User's Manual

32

# RH850/F1K Group

## User's Manual: Hardware

Renesas microcontroller
RH850 Family

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics
www.renesas.com

Rev.1.10 Nov, 2016

# Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

# Notes for CMOS devices

**(1) Voltage application waveform at input pin:**

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between VIL (MAX) and VIH (MIN) due to noise, etc., the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between VIL (MAX) and VIH (MIN).

**(2) Handling of unused input pins:**

Unconnected CMOS device inputs can be cause of malfunction. If an input pin is unconnected, it is possible that an internal input level may be generated due to noise, etc., causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using pull-up or pull-down circuitry. Each unused pin should be connected to power supply or GND via a resistor if there is a possibility that it will be an output pin. All handling related to unused pins must be judged separately for each device and according to related specifications governing the device.

**(3) Precaution against ESD:**

A strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it when it has occurred. Environmental control must be adequate. When it is dry, a humidifier should be used. It is recommended to avoid using insulators that easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors should be grounded. The operator should be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with mounted semiconductor devices.

**(4) Status before initialization:**

Power-on does not necessarily define the initial status of a MOS device. Immediately after the power source is turned ON, devices with reset functions have not yet been initialized. Hence, power-on does not guarantee output pin levels, I/O settings or contents of registers. A device is not initialized until the reset signal is received. A reset operation must be executed immediately after power-on for devices with reset functions.

**(5) Power ON/OFF sequence:**

In the case of a device that uses different power supplies for the internal operation and external interface, as a rule, switch on the external power supply after switching on the internal power supply. When switching the power supply off, as a rule, switch off the external power supply and then the internal power supply. Use of the reverse power on/off sequences may result in the application of an overvoltage to the internal elements of the device, causing malfunction and degradation of internal elements due to the passage of an abnormal current. The correct power on/off sequence must be judged separately for each device and according to related specifications governing the device.

**(6) Input of signal during power off state:**

Do not input signals or an I/O pull-up power supply while the device is not powered. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Input of signals during the power off state must be judged separately for each device and according to related specifications governing the device.

# How to Use This Manual

**Readers** This manual is intended for users who wish to understand the functions of the RH850/F1K and design application systems using the following RH850/F1K microcontrollers:

**Purpose** This manual is intended to give users an understanding of the hardware functions of the RH850/F1K shown in the *Organization* below.

**Organization** This manual is divided into two parts: Hardware (this manual) and Architecture (RH850G3KH User's Manual: Software).

| Hardware | Software |
|---|---|
| Pin functions | Overview |
| CPU function | Processor Model |
| On-chip peripheral functions | Register Reference |
| Flash memory programming | Exceptions and Interrupts |
| | Memory Management |
| | Instruction Reference |
| | Reset |
| | Appendix |

**How to read this manual** It is assumed that the readers of this manual have general knowledge in the fields of electrical engineering, logic circuits, and microcontrollers.

To understand the overall functions of the RH850/F1K.

→ Read this manual according to the Contents.

To understand the details of an instruction function

→ See RH850G3KH User's Manual: Software (R01US0165E) available separately.

**Conventions**

Data significance: Higher digits on the left and lower digits on the right

Active low representation: xxx (overscore over pin or signal name)

Memory map address: Higher addresses on the top and lower addresses on the bottom

Note: Footnote for item marked with Note in the text

Caution: Information requiring particular attention

Remark: Supplementary information

Numeric representation: Binary ... xxxx or $xxxx_B$

Decimal ... xxxx

Hexadecimal ... $xxxx_H$

Prefix indicating power of 2 (address space, memory capacity):

K (kilo): $2^{10} = 1,024$

M (mega): $2^{20} = 1,024^2$

G (giga): $2^{30} = 1,024^3$

# Description of Registers

Each register description includes register access, register address, and register value after a reset, a bit chart, illustrating the arrangement of bits, and a table of bits, describing the meaning of the bit settings.

The standard format for bit charts and tables are described below.

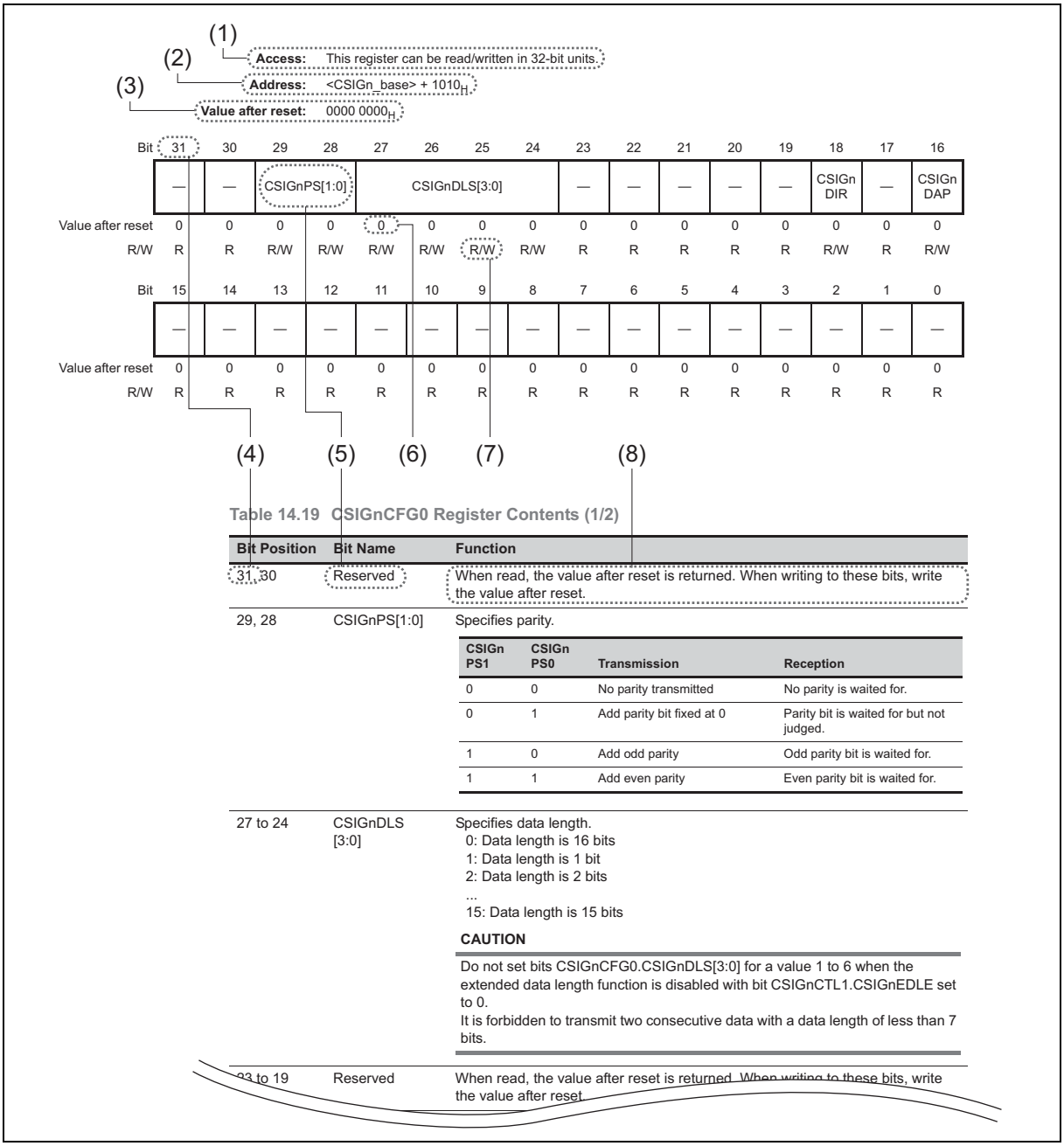



**Table 14.19 CSIGnCFG0 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset is returned. When writing to these bits, write the value after reset. |
| 29, 28 | CSIGnPS[1:0] | Specifies parity. |

| CSIGn PS1 | CSIGn PS0 | Transmission | Reception |
|---|---|---|---|
| 0 | 0 | No parity transmitted | No parity is waited for. |
| 0 | 1 | Add parity bit fixed at 0 | Parity bit is waited for but not judged. |
| 1 | 0 | Add odd parity | Odd parity bit is waited for. |
| 1 | 1 | Add even parity | Even parity bit is waited for. |

| | | |
|---|---|---|
| 27 to 24 | CSIGnDLS [3:0] | Specifies data length.<br>0: Data length is 16 bits<br>1: Data length is 1 bit<br>2: Data length is 2 bits<br>...<br>15: Data length is 15 bits<br>**CAUTION**<br>Do not set bits CSIGnCFG0.CSIGnDLS[3:0] for a value 1 to 6 when the extended data length function is disabled with bit CSIGnCTL1.CSIGnEDLE set to 0.<br>It is forbidden to transmit two consecutive data with a data length of less than 7 bits. |
| 23 to 19 | Reserved | When read, the value after reset is returned. When writing to these bits, write the value after reset. |

(1) Access

The register can be accessed in the bit unit indicated here.

(2) Address

This is the register address.
For base address, see description of base address in each section.

(3) Value after a reset (in hexadecimal notation)

This is the value of all bits of the register after a reset. Values for bytes are given as numbers in the range from 0 to 9 and letters from A to F or as X where they are undefined.

(4) Bit position

This is the bit number.
The bits are numbered from 31 to 0 for 32-bit registers, 15 to 0 for 16-bit registers, and 7 to 0 for 8-bit registers.

(5) Bit name

Bit name or field name is indicated.
When clearly identifying the digits of a bit field is required, do so by using a form such as CSIGnDLS[3:0] above.
Indicate reserved bits by using a dash (—).

(6) Value after a reset (in binary notation)

This is the bit values after a reset.

| | | |
|---|---|---|
| 0 | : | The value after a reset is 0. |
| 1 | : | The value after a reset is 1. |
| — | : | The value after a reset is undefined. |

(7) R/W

This is the bit attribute of all bits of the register.

| | | |
|---|---|---|
| R/W | : | The bit or field is readable and writable. |
| R | : | The bit or field is readable.<br>Note that all reserved bits are indicated as R. When written, the value specified in the bit chart or the value after a reset should be written. |
| W | : | This bit or field is writable. When read, the value is undefined. If a value is indicated in the bit chart, the value is returned. |

(8) Function

This is function of the bit.

All trademarks and registered trademarks are the property of their respective owners.

# Table of Contents


Section 1 Overview ..... 71

1.1 RH850/F1K Product Features ..... 71
1.2 RH850/F1K Functions ..... 72
1.3 RH850/F1K Product Lineup ..... 74
1.4 RH850/F1K Product Block Diagrams ..... 76

Section 2 Pin Function ..... 79

2.1 Pin Connection Diagram ..... 79
2.2 Pin Description ..... 94
2.3 Pin Functions During and After Reset ..... 98
2.4 Port state in Standby Mode ..... 99
2.5 Recommended Connection of Unused Pins ..... 99
2.6 RH850/F1K Port Features ..... 101
2.6.1 Port Group ..... 101
2.6.2 Port Group Index n ..... 101
2.6.3 Register Base Address ..... 101
2.6.4 Clock Supply ..... 101
2.7 Port Functions ..... 102
2.7.1 Functional Overview ..... 102
2.7.2 Terms ..... 103
2.7.2.1 JTAG Ports ..... 103
2.7.3 Overview of Pin Functions ..... 104
2.7.4 Pin Data Input/Output ..... 106
2.7.4.1 Output Data ..... 106
2.7.4.2 Input Data ..... 106
2.7.4.3 Writing to the Pn Register ..... 107
2.8 Schematic View of Port Control ..... 108
2.9 Port Group Configuration Registers ..... 109
2.9.1 Overview ..... 109
2.9.2 Pin Function Configuration ..... 112
2.9.2.1 PMCn / JPMC0 — Port Mode Control Register ..... 112
2.9.2.2 PMCSRn / JPMCSR0 — Port Mode Control Set/Reset Register ..... 113
2.9.2.3 PIPCn — Port IP Control Register ..... 114
2.9.2.4 PMn / APMn / JPM0 — Port Mode Register ..... 115
2.9.2.5 PMSRn / APMSRn / JPMSR0 — Port Mode Set/Reset Register ..... 116
2.9.2.6 PIBCn / APIBCn / JPIBC0 / IPIBC0 — Port Input Buffer Control Register ..... 118
2.9.2.7 PFCn / JPFC0 — Port Function Control Register ..... 119
2.9.2.8 PFCEn / JPFCE0 — Port Function Control Expansion Register ..... 120
2.9.2.9 PFCAEn — Port Function Control Additional Expansion Register ..... 121
2.9.3 Pin Data Input/Output ..... 123
2.9.3.1 PBDCn / APBDCn / JPBDC0 — Port Bidirection Control Register ..... 123
2.9.3.2 PPRn / APPRn / JPPR0 / IPPR0 — Port Pin Read Register ..... 124
2.9.3.3 Pn / APn / JP0 — Port Register ..... 125
2.9.3.4 PNOTn / APNOTn / JPNOT0 — Port NOT Register ..... 126
2.9.3.5 PSRn / APSRn / JPSR0 — Port Set/Reset Register ..... 127

2.9.4 Configuration of Electrical Characteristics ..... 128
2.9.4.1 PUn / JPU0 — Pull-Up Option Register ..... 128
2.9.4.2 PDn / JPD0 — Pull-Down Option Register ..... 129
2.9.4.3 PDSCn / JPDSC0 — Port Drive Strength Control Register ..... 130
2.9.4.4 PODCn / JPODC0 — Port Open Drain Control Register ..... 131
2.9.4.5 PISn/JPIS0 — Port Input Buffer Selection Register ..... 132
2.9.4.6 JPISA0 — Port Input Buffer Selection Advanced Register ..... 133
2.9.5 Port Register Protection ..... 133
2.9.6 Flowchart Examples for Port Settings ..... 134
2.9.6.1 Batch Setting ..... 134
2.9.6.2 Individual Settings ..... 135
2.10 Port (General I/O) Function Overview ..... 138
2.10.1 JTAG Port 0 (JP0) ..... 139
2.10.1.1 Alternative Function ..... 139
2.10.1.2 Control Registers ..... 140
2.10.2 Port 0 (P0) ..... 142
2.10.2.1 Alternative Function ..... 142
2.10.2.2 Control Registers ..... 143
2.10.3 Port 1 (P1) ..... 144
2.10.3.1 Alternative Function ..... 144
2.10.3.2 Control Registers ..... 145
2.10.4 Port 2 (P2) ..... 147
2.10.4.1 Alternative Function ..... 147
2.10.4.2 Control Registers ..... 148
2.10.5 Port 8 (P8) ..... 149
2.10.5.1 Alternative Function ..... 149
2.10.5.2 Control Registers ..... 150
2.10.6 Port 9 (P9) ..... 151
2.10.6.1 Alternative Function ..... 151
2.10.6.2 Control Registers ..... 152
2.10.7 Port 10 (P10) ..... 153
2.10.7.1 Alternative Function ..... 153
2.10.7.2 Control Registers ..... 154
2.10.8 Port 11 (P11) ..... 155
2.10.8.1 Alternative Function ..... 155
2.10.8.2 Control Registers ..... 156
2.10.9 Port 12 (P12) ..... 158
2.10.9.1 Alternative Function ..... 158
2.10.9.2 Control Registers ..... 159
2.10.10 Port 18 (P18) ..... 161
2.10.10.1 Alternative Function ..... 161
2.10.10.2 Control Registers ..... 162
2.10.11 Port 20 (P20) ..... 164
2.10.11.1 Alternative Function ..... 164
2.10.11.2 Control Registers ..... 165
2.10.12 Analog Port 0 (AP0) ..... 167
2.10.12.1 Alternative Function ..... 167
2.10.12.2 Control Registers ..... 168

2.10.13 Analog Port 1 (AP1) ..... 169
2.10.13.1 Alternative Function ..... 169
2.10.13.2 Control Registers ..... 170
2.10.14 Input Port 0 (IP0) ..... 171
2.10.14.1 Alternative Function ..... 171
2.10.14.2 Control Registers ..... 172
2.11 Port (Special I/O) Function Overview ..... 173
2.11.1 Special I/O after Reset ..... 173
2.11.1.1 P8_6: $\overline{\text{RESETOUT}}$ ..... 173
2.11.1.2 JP0_0 to JP0_6: Debug Interface ..... 175
2.11.1.3 FPDR(JP0_0), FPDT(JP0_1), FPCK(JP0_2): Flash Programmer ..... 175
2.11.1.4 Mode Pins ..... 175
2.11.1.5 IP0_0: XT2 ..... 175
2.11.2 A/D Input Alternative I/O ..... 176
2.11.3 Special I/O Control ..... 178
2.11.3.1 Direct I/O Control (PIPC) ..... 178
2.11.3.2 Input Buffer Control (PISn/JPIS0, JPISA0) ..... 179
2.11.3.3 Output Buffer Control (PDSC) ..... 183
2.12 Noise Filter & Edge/Level Detector ..... 186
2.12.1 Port Filter Assignment ..... 186
2.12.1.1 Input Pins that Incorporate Analog Filter Type A ..... 186
2.12.1.2 Input Pins that Incorporate Analog Filter Type B ..... 187
2.12.1.3 Input Pins that Incorporate Analog Filter Type C ..... 187
2.12.1.4 Input Pins that Incorporate Digital Filter Type D ..... 188
2.12.1.5 Input Pins that Incorporate Digital Filter Type E ..... 190
2.12.2 Clock Supply for Port Filters ..... 194
2.13 Description of Port Noise Filter & Edge/Level Detection ..... 195
2.13.1 Overview ..... 195
2.13.1.1 Analog Filter Types ..... 195
2.13.1.2 Digital Filter Types ..... 195
2.13.2 Analog Filters ..... 196
2.13.2.1 Analog Filter Characteristic ..... 196
2.13.2.2 Analog Filter Control Registers ..... 196
2.13.2.3 Analog Filter in Standby Mode ..... 196
2.13.3 Digital Filters ..... 199
2.13.3.1 Digital Filter Characteristic ..... 199
2.13.3.2 Digital Filter Groups ..... 200
2.13.3.3 Digital Filters in Standby Mode ..... 200
2.13.3.4 Digital Filter Control Registers ..... 201
2.13.4 Filter Control Registers ..... 202
2.13.4.1 FCLA0CTLm_<name> — Filter Control Register ..... 203
2.13.4.2 DNFA<name>CTL — Digital Noise Elimination Control Register ..... 204
2.13.4.3 DNFA<name>EN — Digital Noise Elimination Enable Register ..... 205
2.13.4.4 DNFA<name>ENH – Digital Noise Elimination Enable H Register ..... 206
2.13.4.5 DNFA<name>ENL – Digital Noise Elimination Enable L Register ..... 207

Section 3 CPU System ..... 208
3.1 Overview ..... 208
3.1.1 Block Configuration ..... 208
3.2 CPU ..... 210
3.2.1 Core Functions ..... 210
3.2.1.1 Features ..... 210
3.2.1.2 Register Set ..... 211
3.2.1.3 Instruction ..... 246
3.2.2 Buffers for Code Flash ..... 247
3.2.2.1 Features ..... 247
3.2.2.2 Function of Buffers ..... 247
3.2.2.3 Registers for Buffer Control ..... 248
3.2.3 Reliability Functions ..... 249
3.2.3.1 PE Guard Function (PEG) ..... 249
3.2.3.2 System Error Generator Function (SEG) ..... 255
3.3 Notes ..... 260
3.3.1 Synchronization of Store Instruction Completion and Subsequent Instruction Execution 260
3.3.2 Ensure Coherency after Rewriting the Code Flash ..... 261
3.3.3 Access to Registers by Using Bit-Manipulation Instructions ..... 261
3.3.4 Caution of Prefetching ..... 261

Section 4 Address Space ..... 262
4.1 Address Space ..... 262
4.2 Address Space Viewed from Each Bus Master ..... 265
4.2.1 Space in which Instructions can be Fetched ..... 265
4.2.2 Data Space Accessible by CPU ..... 265
4.2.3 Data Space Accessible by DMA ..... 265
4.3 Peripheral I/O Address Map ..... 266

Section 5 Write-Protected Registers ..... 271
5.1 Overview ..... 271
5.1.1 Functional Overview ..... 271
5.1.2 Writing Procedure to Write-Protected Registers ..... 271
5.1.3 Interrupt during Write Protection Unlock ..... 272
5.1.4 Emulation Break during Write Protection Unlock Sequence ..... 273
5.1.5 Write-Protection Target Registers ..... 273
5.2 Registers ..... 276
5.2.1 List of Registers ..... 276
5.2.2 Details of Control Protection Cluster Registers ..... 278
5.2.2.1 PROTCMDn — Protection Command Register ..... 278
5.2.2.2 PROTSn — Protection Status Register ..... 279
5.2.3 Details of Clock Monitor Protection Cluster Registers ..... 280
5.2.3.1 CLMAnPCMD — CLMAn Protection Command Register ..... 280
5.2.3.2 CLMAnPS — CLMAn Protection Status Register ..... 281
5.2.3.3 PROTCMDCLMA — Clock Monitor Test Protection Command Register ..... 282

5.2.3.4 PROTSCLMA — Clock Monitor Test Protection Status Register ... 283
5.2.4 Details of Core Voltage Monitor Protection Cluster Registers ... 284
5.2.4.1 PROTCMDCVM — Core Voltage Monitor Protection Command Register ... 284
5.2.4.2 PROTSCVM — Core Voltage Monitor Protection Status Register ... 285
5.2.5 Details of Port Protection Cluster Registers ... 286
5.2.5.1 PPCMDn — Port Protection Command Register ... 286
5.2.5.2 PPROTSn — Port Protection Status Register ... 287
5.2.6 Details of Self-Programming Protection Cluster Registers ... 288
5.2.6.1 FLMDPCMD — FLMD Protection Command Register ... 288
5.2.6.2 FLMDPS — FLMD Protection Error Status Register ... 289

Section 6 Operating Mode ... 290

Section 7 Exception/Interrupts ... 291

7.1 Features ... 291
7.2 Interrupt Sources ... 293
7.2.1 Interrupt Sources ... 293
7.2.1.1 FE Level Non-Maskable Interrupts ... 293
7.2.1.2 FE Level Maskable Interrupts ... 294
7.2.1.3 EI Level Maskable Interrupts ... 295
7.2.2 List of Registers ... 307
7.2.3 FE Level Non-Maskable Interrupt Sources ... 308
7.2.3.1 WDTNMIF — FENMI Factor Register ... 308
7.2.3.2 WDTNMIFC — WDTNMI Factor Clear Register ... 309
7.2.4 FE Level Maskable Interrupt Sources ... 310
7.2.4.1 FEINTF — FEINT Factor Register ... 310
7.2.4.2 FEINTFMSK — FEINT Factor Mask Register ... 312
7.2.4.3 FEINTFC — FEINT Factor Clear Register ... 314
7.3 Edge/Level Detection ... 316
7.4 Interrupt Controller Control Registers ... 317
7.4.1 List of Registers ... 317
7.4.2 ICxxx — EI Level Interrupt Control Registers ... 318
7.4.3 IMRm — EI Level Interrupt Mask Registers (m = 0 to 11) ... 320
7.4.4 FNC — FE Level NMI Status Register ... 321
7.4.5 FIC — FE Level Maskable Interrupt Status Register ... 322
7.5 EI Level Maskable Interrupt Select Registers ... 323
7.5.1 List of Registers ... 323
7.5.2 SELB_INTC1 — INTC1 Interrupt Select Register ... 324
7.5.3 SELB_INTC2 — INTC2 Interrupt Select Register ... 326
7.6 Interrupt Function System Registers ... 327
7.6.1 FPIPR — FPI Exception Interrupt Priority ... 327
7.6.2 ISPR — Priority of Interrupt being Serviced ... 327
7.6.3 PMR — Interrupt Priority Masking ... 327
7.6.4 ICSR — Interrupt Control Status ... 327
7.6.5 INTCFG — Interrupt Function Setting ... 327
7.7 Operation when Acknowledging an Interrupt ... 328

7.7.1 Exception Source Codes for Different Types of SYSERR Exceptions ..... 330
7.8 Return from Interrupts ..... 331
7.9 Interrupt Operation ..... 332
7.9.1 Interrupt Mask Function of EI Level Maskable Interrupt (EIINT) ..... 332
7.9.2 Interrupt Priority Level Judgment ..... 332
7.9.2.1 Comparison with the Priority Level of the Interrupt Currently being Handled ..... 333
7.9.2.2 Masking through Priority Mask Register (PMR) ..... 333
7.9.2.3 The Requested Interrupt Source with the Highest Priority Level is Selected ..... 333
7.9.2.4 Interrupt Suspended by CPU ..... 333
7.9.3 Interrupt Request Acknowledgement Conditions and the Priority ..... 336
7.9.4 Exception Priority of Interrupts and the Priority Mask ..... 336
7.9.5 Interrupt Priority Mask ..... 336
7.9.6 Priority Mask Function ..... 337
7.9.7 Exception Management ..... 337
7.10 Exception Handler Address ..... 338
7.10.1 Direct Vector Method ..... 339
7.10.2 Table Reference Method ..... 341

Section 8 DMA Controller ..... 343

8.1 Features of RH850/F1K DMA Controller ..... 343
8.1.1 Number of Channels ..... 343
8.1.2 Register Base Address ..... 343
8.1.3 Interrupt Requests ..... 344
8.1.4 DMA Trigger Factors ..... 345
8.2 Overview ..... 349
8.2.1 Overview ..... 349
8.2.2 Term Definition ..... 350
8.3 DMA Function ..... 351
8.3.1 Basic Operation of DMA Transfer ..... 351
8.3.1.1 Transfer Mode ..... 351
8.3.1.2 Executing a DMA Cycle ..... 351
8.3.1.3 Updating Transfer Information ..... 351
8.3.1.4 Last Transfer and Address Reload Transfer ..... 352
8.3.1.5 Transfer Completion Interrupt Output ..... 352
8.3.1.6 Continuous Transfer ..... 353
8.3.2 Channel Priority Order ..... 355
8.3.2.1 DMAC Channel Arbitration ..... 355
8.3.2.2 Interface Arbitration ..... 356
8.3.3 Reload Function ..... 357
8.3.3.1 Overview of the Reload Function ..... 357
8.3.3.2 Operation of Reload Function 1 ..... 357
8.3.3.3 Reload Function 2 ..... 358
8.3.3.4 Timing of Setting DMAC Reload Registers ..... 360
8.3.4 Chain Function ..... 361
8.3.4.1 Overview ..... 361
8.3.4.2 Setting Up the Chain Function ..... 362

8.3.4.3 Caution for Using the Chain Function ... 362
8.3.5 DMAC Operation ... 363
8.3.5.1 Types of DMA Transfer Requests and Assigning DMA Transfer Requests ... 363
8.3.5.2 Generating and Accepting a Software DMA Transfer Request ... 363
8.3.5.3 Executing DMA Transfer ... 363
8.4 Suspension, Resume, Transfer Abort, and Clearing of a DMA Transfer Request ... 364
8.4.1 DMA Suspension and Resume by Software Control ... 364
8.4.2 Suspension, Resume, and Transfer Abort of a DMAC Channel ... 364
8.4.3 Masking and Clearing a Hardware DMA Transfer Request by the DTFR ... 365
8.4.4 List of Suspend, Resume, and Transfer Abort Functions ... 366
8.5 Error Control ... 367
8.5.1 Type of Error ... 367
8.5.2 DMA Transfer Error ... 367
8.5.2.1 Operation of a DMAC When DMA Transfer Error Occurs ... 367
8.6 Reliability Functions ... 368
8.6.1 Overview ... 368
8.6.2 Register Access Protection Function ... 368
8.6.2.1 Identifying the Accessing Master ... 368
8.6.2.2 Master Access ... 368
8.6.2.3 Channel Assignment ... 368
8.6.2.4 Illegal Access ... 369
8.6.3 Master Information Inherit Function ... 370
8.6.4 Other Reliability Functions ... 370
8.6.4.1 Restriction on the Next Channel in the Chain ... 370
8.7 Setting Up DMA Transfer ... 371
8.7.1 Overview of Setting Up DMA ... 371
8.7.2 Setting Up the Overall DMA Operation ... 372
8.7.3 Setting Up the DMA Channel Setting ... 372
8.7.3.1 Setting Up the DMAC Channel Setting ... 372
8.8 Global Registers ... 374
8.8.1 List of Global Register Address ... 374
8.8.2 Details of Global Registers ... 375
8.8.2.1 DMACTL — DMA Control Register ... 375
8.8.2.2 DMACER — DMAC Error Register ... 376
8.8.2.3 DM0CMV — DMAC0 Register Access Protection Violation Register ... 377
8.8.2.4 DM1CMV — DMAC1 Register Access Protection Violation Register ... 378
8.8.2.5 CMVC — Register Access Protection Violation Clear Register ... 379
8.8.2.6 DMniCM — DMAC Channel Master Setting (ni = 00 to 07 and 10 to 17) ... 380
8.9 DMAC Channel Registers ... 381
8.9.1 DMAC Channel Register Addresses ... 381
8.9.2 Details of DMAC Channel Registers ... 382
8.9.2.1 DSAm — DMAC Source Address Register ... 382
8.9.2.2 DDAm — DMAC Destination Address Register ... 383
8.9.2.3 DTCm — DMAC Transfer Count Register ... 384
8.9.2.4 DTCTm — DMAC Transfer Control Register ... 385
8.9.2.5 DRSAm — DMAC Reload Source Address Register ... 388

8.9.2.6 DRDAm — DMAC Reload Destination Address Register ... 389
8.9.2.7 DRTCm — DMAC Reload Transfer Count Register ... 390
8.9.2.8 DCENm — DMAC Channel Operation Enable Setting Register ... 391
8.9.2.9 DCSTm — DMAC Transfer Status Register ... 392
8.9.2.10 DCSTSm — DMAC Transfer Status Set Register ... 394
8.9.2.11 DCSTCm — DMAC Transfer Status Clear Register ... 395
8.9.2.12 DTFRm — DTFR Setting Register ... 396
8.9.2.13 DTFRRQm — DTFR Transfer Request Status Register ... 397
8.9.2.14 DTFRRQCm — DTFR Transfer Request Clear Register ... 398

Section 9 Reset Controller ... 399

9.1 Overview ... 399
9.1.1 Reset Sources ... 399
9.1.2 Reset Controller Redundancy ... 400
9.1.3 Reset Output ($\overline{\text{RESETOUT}}$) ... 401
9.1.4 Reset Flag ... 401
9.1.5 Clock Supply ... 401
9.2 Configuration ... 402
9.2.1 Block Diagram ... 402
9.3 Registers ... 405
9.3.1 Reset Controller Registers Overview ... 405
9.3.2 Details of Reset Flag Registers ... 406
9.3.2.1 RESF — Reset Factor Register ... 406
9.3.2.2 RESFC — Reset Factor Clear Register ... 408
9.3.2.3 RESFR — Redundant Reset Factor Register ... 409
9.3.2.4 RESFCR — Redundant Reset Factor Clear Register ... 411
9.3.3 Details of Software Reset Control Registers ... 412
9.3.3.1 SWRESA — Software Reset Register ... 412
9.3.4 Details of Cyclic RUN Mode Reset Vector Address Register ... 413
9.3.4.1 CYCRBASE — Cyclic RUN mode RBASE Register ... 413
9.4 Functional Description ... 414
9.4.1 Reset Flags ... 414
9.4.2 Power-On-Clear (POC) Reset ... 414
9.4.3 Low-Voltage Indicator (LVI) Reset ... 417
9.4.4 Core Voltage Monitor (CVM) Reset ... 418
9.4.5 External Reset ($\overline{\text{RESET}}$) ... 419
9.4.6 Watchdog Timer (WDTA) Reset ... 420
9.4.7 Software Reset ... 420
9.4.8 Clock Monitor (CLMA) Reset ... 421
9.4.9 Debugger Reset ... 421
9.4.10 Reset Vector Address of CPU ... 421

Section 10 Power Supply Circuit ... 422

10.1 Function ... 422
10.1.1 Power Supply Pins ... 423
10.1.2 Block Diagram of Power Domains ... 424

10.1.3 Power Domains Arrangement ... 426

Section 11 Supply Voltage Monitor ... 427

11.1 Overview ... 427
11.1.1 Functional Overview ... 427
11.1.2 Power-On Clear (POC) ... 428
11.1.3 Low Voltage Indicator Circuit (LVI) ... 428
11.1.3.1 LVI Reference Voltage ... 428
11.1.3.2 LVI Reset (LVIRES) ... 428
11.1.3.3 LVI Interrupt (INTLVIL/INTLVIH) ... 429
11.1.3.4 LVI Setting Procedure ... 430
11.1.3.5 Clock Supply to the LVI ... 430
11.1.4 Core Voltage Monitor (CVM) ... 431
11.1.4.1 CVM Reset ($\overline{\text{CVMRES}}$) ... 431
11.1.4.2 CVM Setting ... 431
11.1.4.3 Diagnostic (DIAG) Mode ... 431
11.1.4.4 Clock Supply to the CVM ... 432
11.1.5 RAM Retention Voltage Indicator (Very-Low-Voltage Detection Circuit, VLVI) ... 433
11.1.5.1 Clock Supply to the VLVI ... 433
11.1.5.2 Retention RAM Content Retention ... 433
11.1.6 Block Diagram ... 434
11.2 Registers ... 435
11.2.1 List of Registers ... 435
11.2.2 Low-Voltage Indicator Reset Control Registers ... 436
11.2.2.1 LVICNT — LVI Control Register ... 436
11.2.3 Core Voltage Monitor Control Registers ... 437
11.2.3.1 CVMF — CVM Factor Register ... 437
11.2.3.2 CVMDE — CVM Detection Enable Register ... 438
11.2.3.3 CVMDIAG — CVM Diagnostic Mode Setting Register ... 439
11.2.4 Very-Low-Voltage Detection Control Registers ... 440
11.2.4.1 VLVF — Very-Low-Voltage Detection Register ... 440
11.2.4.2 VLVFC — Very-Low-Voltage Detection Clear Register ... 441

Section 12 Clock Controller ... 442

12.1 Features of Clock Controller of RH850/F1K ... 442
12.2 Configuration of Clock Controller ... 444
12.2.1 Clock Generation Circuits ... 446
12.2.2 Clock Selection ... 447
12.2.3 Clock Domains ... 448
12.2.4 Resetting Clock Oscillators ... 448
12.3 Clock Oscillators ... 449
12.3.1 Main Oscillator (MainOSC) ... 449
12.3.2 Sub Oscillator (SubOSC) ... 451
12.3.3 High Speed Internal Oscillator (HS IntOSC) ... 452
12.3.4 Low Speed Internal Oscillator (LS IntOSC) ... 454
12.3.5 PLL ... 455

12.3.5.1 PLL Parameters ... 457
12.4 Registers ... 458
12.4.1 List of Registers ... 458
12.4.2 Clock Oscillator Registers ... 460
12.4.2.1 MOSCE — MainOSC Enable Register ... 460
12.4.2.2 MOSCS — MainOSC Status Register ... 462
12.4.2.3 MOSCC — MainOSC Control Register ... 463
12.4.2.4 MOSCST — MainOSC Stabilization Time Register ... 464
12.4.2.5 MOSCSTPM — MainOSC Stop Mask Register ... 465
12.4.2.6 MOSCM — MainOSC Mode Control Register ... 466
12.4.2.7 SOSCE — SubOSC Enable Register ... 467
12.4.2.8 SOSCS — SubOSC Status Register ... 468
12.4.2.9 SOSCST — SubOSC Stabilization Time Register ... 469
12.4.2.10 ROSCE — HS IntOSC Enable Register ... 470
12.4.2.11 ROSCS — HS IntOSC Status Register ... 471
12.4.2.12 ROSCSTPM — HS IntOSC Stop Mask Register ... 472
12.4.2.13 PLLE — PLL Enable Register ... 473
12.4.2.14 PLLS — PLL Status Register ... 474
12.4.2.15 PLLC — PLL Control Register ... 475
12.4.2.16 PLL Input Clock Selection ... 479
12.4.2.17 PPLLCLK Source Clock Selection ... 481
12.4.2.18 ROSCUT — HS IntOSC User Trimming Register ... 483
12.4.3 Clock Selector Control Register ... 484
12.4.3.1 WDTA0 Clock Domain C_AWO_WDTA ... 484
12.4.3.2 TAUJ Clock Domain C_AWO_TAUJ ... 487
12.4.3.3 RTCA Clock Domain C_AWO_RTCA ... 492
12.4.3.4 ADCA0 Clock Domain C_AWO_ADCA ... 497
12.4.3.5 FOUT Clock Domain C_AWO_FOUT ... 502
12.4.3.6 CPU Clock Domain C_ISO_CPUCLK ... 505
12.4.3.7 Peripheral Clock Domains C_ISO_PERI1 and C_ISO_PERI2 ... 509
12.4.3.8 RLIN Clock Domains C_ISO_LIN ... 513
12.4.3.9 ADCA1 Clock Domain C_ISO_ADCA ... 518
12.4.3.10 RS-CAN Clock Domains C_ISO_CAN and C_ISO_CANOSC ... 522
12.4.3.11 CSI Clock Domain C_ISO_CSI ... 528
12.4.3.12 IIC Clock Domain C_ISO_IIC ... 530
12.5 Clock Domain Setting Method ... 532
12.5.1 Clock Domain Setting ... 532
12.5.1.1 Overview of Clock Selector Register ... 532
12.5.1.2 Setting Procedure for Clock Domain ... 533
12.5.2 Stopping the Clock in Stand-by Mode ... 533
12.5.3 Clock Domain Settings ... 534
12.6 Frequency Output Function (FOUT) ... 536
12.6.1 Functional Overview ... 536
12.6.2 Clock Supply ... 536
12.6.3 Registers ... 537
12.6.3.1 List of Registers ... 537
12.6.3.2 FOUTDIV — Clock Division Ratio Register ... 538
12.6.3.3 FOUTSTAT — Clock Divider Status Register ... 539

Section 13 Clock Monitor (CLMA) ..... 540
13.1 Features of RH850/F1K CLMA ..... 540
13.1.1 Number of Channels ..... 540
13.1.2 Register Base Addresses ..... 540
13.1.3 Clock Supply to CLMA ..... 540
13.1.4 Reset Sources ..... 541
13.1.5 Internal Input/Output Signals ..... 541
13.2 Overview ..... 542
13.2.1 Functional Overview ..... 542
13.3 Enabling CLMA ..... 543
13.4 Functions ..... 544
13.4.1 Detection of Abnormal Clock Frequencies ..... 544
13.4.2 Notification of Abnormal Clock Frequency ..... 546
13.5 Registers ..... 547
13.5.1 List of Registers ..... 547
13.5.2 CLMAnCTL0 — CLMAn Control Register 0 ..... 548
13.5.3 CLMAnCMPH — CLMAn Compare Register H ..... 549
13.5.4 CLMAnCMPL — CLMAn Compare Register L ..... 549
13.5.5 CLMATEST — CLMA Test Register ..... 550
13.5.6 CLMATESTS — CLMA Test Status Register ..... 551
13.5.7 CLMAnEMU0 — CLMAn Emulation Register 0 ..... 552
13.6 Usage Notes for CLMAn ..... 553
Section 14 Stand-By Controller (STBC) ..... 554
14.1 Functions ..... 554
14.1.1 Types of Stand-By Mode ..... 554
14.1.2 Wake-Up Control ..... 555
14.1.2.1 Wake-Up Factors for Stand-By Modes ..... 555
14.1.2.2 Setting of Wake-Up Factors ..... 557
14.1.3 On-Chip Debug Wake-Up ..... 560
14.1.4 I/O Buffer Control ..... 561
14.1.4.1 I/O Buffer Hold State ..... 561
14.1.4.2 I/O Buffers during STOP Mode ..... 561
14.1.4.3 I/O Buffers during DeepSTOP Mode ..... 561
14.1.5 Transition to Stand-By Mode ..... 562
14.1.6 Clock Supply ..... 562
14.2 Registers ..... 563
14.2.1 List of Registers ..... 563
14.2.2 Details of Stand-By Controller Control Registers ..... 564
14.2.2.1 STBC0PSC — Power Save Control Register ..... 564
14.2.2.2 STBC0STPT — Power Stop Trigger Register ..... 565
14.2.2.3 WUF0/WUF20/WUF_ISO0 — Wake-Up Factor Registers ..... 566
14.2.2.4 WUFMSK0/WUFMSK20/WUFMSK_ISO0 — Wake-Up Factor Mask Registers ..... 567
14.2.2.5 WUFC0/WUFC20/WUFC_ISO0 — Wake-Up Factor Clear Registers ..... 568
14.2.2.6 IOHOLD — I/O Buffer Hold Control Register ..... 569

14.3 Mode Transition ..... 570
14.3.1 STOP Mode ..... 570
14.3.2 DeepSTOP Mode ..... 572
14.3.3 Cyclic RUN Mode ..... 575
14.3.4 Cyclic STOP Mode ..... 577
14.4 Writing to the Stand-By Controller Related Registers ..... 578
14.5 Clock Oscillator Behavior During Stand-By Mode Transition ..... 578
14.6 Cautions when Using Stand-By Modes ..... 584
14.6.1 Cautions Concerning Transitioning to DeepSTOP Mode When Using a Debugger ..... 584

Section 15 Low-Power Sampler (LPS) ..... 585

15.1 Features of RH850/F1K LPS ..... 585
15.1.1 Number of Units ..... 585
15.1.2 Register Base Address ..... 586
15.1.3 Clock Supply ..... 586
15.1.4 Interrupt Request ..... 586
15.1.5 Reset Sources ..... 587
15.1.6 External Input/Output Signals ..... 587
15.1.7 Internal Input/Output Signals ..... 587
15.2 Overview ..... 588
15.2.1 Functional Overview ..... 588
15.3 Registers ..... 589
15.3.1 List of Registers ..... 589
15.3.2 SCTLR — LPS Control Register ..... 590
15.3.3 EVFR — Event Flag Register ..... 592
15.3.4 DPSELR0 — DPIN Select Register 0 ..... 593
15.3.5 DPSELRM — DPIN Select Register M ..... 594
15.3.6 DPSELRH — DPIN Select Register H ..... 595
15.3.7 DPDSR0 — DPIN Data Set Register 0 ..... 596
15.3.8 DPDSRM — DPIN Data Set Register M ..... 597
15.3.9 DPDSRH — DPIN Data Set Register H ..... 598
15.3.10 DPDIMR0 — DPIN Data Input Monitor Register 0 ..... 599
15.3.11 DPDIMR1 — DPIN Data Input Monitor Register 1 ..... 599
15.3.12 DPDIMR2 — DPIN Data Input Monitor Register 2 ..... 600
15.3.13 DPDIMR3 — DPIN Data Input Monitor Register 3 ..... 600
15.3.14 DPDIMR4 — DPIN Data Input Monitor Register 4 ..... 601
15.3.15 DPDIMR5 — DPIN Data Input Monitor Register 5 ..... 601
15.3.16 DPDIMR6 — DPIN Data Input Monitor Register 6 ..... 602
15.3.17 DPDIMR7 — DPIN Data Input Monitor Register 7 ..... 602
15.3.18 CNTVAL — Count Value Register ..... 603
15.3.19 SOSTR — LPS Operation Status Register ..... 604
15.4 Digital Input Mode ..... 605
15.4.1 Digital Port Error Interrupt ..... 611
15.5 Analog Input Mode ..... 611

15.5.1 Analog Port Error Interrupt ... 616

Section 16 Clocked Serial Interface G (CSIG) ... 617

16.1 Features of RH850/F1K CSIG ... 617
16.1.1 Number of Units ... 617
16.1.2 Register Base Address ... 617
16.1.3 Clock Supply ... 617
16.1.4 Interrupt Request ... 618
16.1.5 Reset Sources ... 618
16.1.6 External Input/Output Signals ... 619
16.1.7 Data Consistency Check ... 619
16.2 Overview ... 620
16.2.1 Functional Overview ... 620
16.2.2 Functional Overview Description ... 620
16.2.3 Block Diagram ... 621
16.3 Registers ... 622
16.3.1 List of Registers ... 622
16.3.2 CSIGnCTL0 — CSIGn Control Register 0 ... 623
16.3.3 CSIGnCTL1 — CSIGn Control Register 1 ... 624
16.3.4 CSIGnCTL2 — CSIGn Control Register 2 ... 626
16.3.5 CSIGnSTR0 — CSIGn Status Register 0 ... 627
16.3.6 CSIGnSTCR0 — CSIGn Status Clear Register 0 ... 629
16.3.7 CSIGnBCTL0 — CSIGn Rx-Only Mode Control Register 0 ... 630
16.3.8 CSIGnCFG0 — CSIGn Configuration Register 0 ... 631
16.3.9 CSIGnTX0W — CSIGn Transmission Register 0 for Word Access ... 633
16.3.10 CSIGnTX0H — CSIGn Transmission Register 0 for Half Word Access ... 634
16.3.11 CSIGnRX0 — CSIGn Reception Register 0 ... 634
16.3.12 CSIGnEMU — CSIGn Emulation Register ... 635
16.3.13 List of Cautions ... 636
16.4 Interrupt Sources ... 637
16.4.1 Interrupt Delay ... 637
16.4.2 INTCSIGTIC (Communication Status Interrupt) ... 638
16.4.3 INTCSIGTIR (Reception Status Interrupt) ... 639
16.4.4 INTCSIGTIRE (Communication Error Interrupt) ... 639
16.5 Operation ... 640
16.5.1 Master/Slave Mode ... 640
16.5.1.1 Master Mode ... 640
16.5.1.2 Slave Mode ... 641
16.5.2 Master/Slave Connections ... 642
16.5.2.1 One Master and One Slave ... 642
16.5.2.2 One Master and Multiple Slaves ... 643
16.5.3 Transmission Clock Selection ... 644
16.5.4 Data Transfer Modes ... 645
16.5.4.1 Transmit-Only Mode ... 645
16.5.4.2 Receive-Only Mode ... 645

16.5.4.3 Transmit/Receive Mode ... 645
16.5.5 Data Length Selection ... 646
16.5.5.1 Data Length Selection without Extended Length ... 646
16.5.5.2 Data Length Selection with Extended Data Length ... 646
16.5.6 Serial Data Direction Selection Function ... 648
16.5.7 Communication in Slave Mode ... 649
16.5.8 Handshake Function ... 650
16.5.8.1 Slave Mode ... 650
16.5.8.2 Master Mode ... 651
16.5.9 Loop-Back Mode ... 652
16.5.10 Error Detection ... 653
16.5.10.1 Data Consistency Check ... 653
16.5.10.2 Parity Check ... 654
16.5.10.3 Overrun Error ... 654
16.6 Operating Procedures ... 656
16.6.1 Master Mode Transmission/Reception by DMA ... 656

Section 17 Clocked Serial Interface H (CSIH) ... 658
17.1 Features of RH850/F1K CSIH ... 658
17.1.1 Number of Units ... 658
17.1.2 Register Base Address ... 659
17.1.3 Clock Supply ... 659
17.1.4 Interrupt Requests ... 660
17.1.5 Reset Sources ... 660
17.1.6 External Input/Output Signals ... 661
17.1.7 Data Consistency Check ... 662
17.2 Overview ... 663
17.2.1 Functional Overview ... 663
17.2.2 Functional Overview Description ... 664
17.2.3 Block Diagram ... 665
17.3 Registers ... 666
17.3.1 List of Registers ... 666
17.3.2 CSIHnCTL0 — CSIHn Control Register 0 ... 667
17.3.3 CSIHnCTL1 — CSIHn Control Register 1 ... 668
17.3.4 CSIHnCTL2 — CSIHn Control Register 2 ... 670
17.3.5 CSIHnSTR0 — CSIHn Status Register 0 ... 672
17.3.6 CSIHnSTCR0 — CSIHn Status Clear Register 0 ... 676
17.3.7 CSIHnMCTL0 — CSIHn Memory Control Register 0 ... 677
17.3.8 CSIHnMCTL1 — CSIHn Memory Control Register 1 ... 678
17.3.9 CSIHnMCTL2 — CSIHn Memory Control Register 2 ... 679
17.3.10 CSIHnMRWP0 — CSIHn Memory Read/Write Pointer Register 0 ... 681
17.3.11 CSIHnCFGx — CSIHn Configuration Register x ... 683
17.3.12 CSIHnTX0W — CSIHn Transmit Data Register 0 for Word Access ... 688
17.3.13 CSIHnTX0H — CSIHn Transmit Data Register 0 for Half Word Access ... 690
17.3.14 CSIHnRX0W — CSIHn Receive Data Register 0 for Word Access ... 691
17.3.15 CSIHnRX0H — CSIHn Receive Data Register 0 for Half Word Access ... 692

17.3.16 CSIHnEMU — CSIHn Emulation Register ..... 692
17.3.17 CSIHnBRSy — CSIHn Baud Rate Setting Register y (y = 0 to 3) ..... 693
17.3.18 List of Cautions ..... 694
17.4 Interrupt Sources ..... 697
17.4.1 Overview ..... 697
17.4.2 Interrupt Delay ..... 698
17.4.3 INTCSIHTIC (Communication Status Interrupt) ..... 699
17.4.3.1 INTCSIHTIC in Direct Access Mode ..... 700
17.4.3.2 INTCSIHTIC in FIFO Mode ..... 701
17.4.3.3 INTCSIHTIC in Job Mode ..... 702
17.4.4 INTCSIHTIR (Reception Status Interrupt) ..... 703
17.4.4.1 INTCSIHTIR in Direct Access Mode ..... 703
17.4.4.2 INTCSIHTIR in Dual Buffer Mode ..... 704
17.4.5 INTCSIHTIRE (Communication Error Interrupt) ..... 705
17.4.6 INTCSIHTIJC (Job Completion Interrupt) ..... 706
17.5 Operation ..... 707
17.5.1 Operating Modes (Master/Slave) ..... 707
17.5.1.1 Master Mode ..... 707
17.5.1.2 Slave Mode ..... 708
17.5.2 Master/Slave Connections ..... 709
17.5.2.1 One Master and One Slave ..... 709
17.5.2.2 One Master and Multiple Slaves ..... 710
17.5.3 Chip Selection (CS) Features ..... 711
17.5.3.1 Configuration Registers ..... 711
17.5.3.2 CS Example ..... 713
17.5.3.3 Job Concept ..... 713
17.5.4 Details of Chip Select Timing ..... 714
17.5.4.1 Changing the Clock Phase ..... 714
17.5.4.2 Changing the Data Phase ..... 716
17.5.5 Transmission Clock Selection ..... 717
17.5.6 CSIH Buffer Memory ..... 719
17.5.6.1 FIFO Mode ..... 719
17.5.6.2 Dual Buffer Mode ..... 720
17.5.6.3 Transmit-Only Buffer Mode ..... 720
17.5.6.4 Direct Access Mode ..... 720
17.5.7 Data Transfer Modes ..... 721
17.5.7.1 Transmit-Only Mode ..... 721
17.5.7.2 Receive-Only Mode ..... 721
17.5.7.3 Transmit/Receive Mode ..... 721
17.5.7.4 Summary ..... 721
17.5.8 Data Length Selection ..... 722
17.5.8.1 Data Length from 2 to 16 Bits ..... 722
17.5.8.2 Data Length Greater than 16 Bits ..... 723
17.5.9 Serial Data Direction Selection ..... 725
17.5.10 Slave Select (SS) Function ..... 726
17.5.10.1 Communication Timing Using SS Function ..... 726
17.5.10.2 CSIHTSSO Operation ..... 727
17.5.11 Handshake Function ..... 728

17.5.11.1 Slave Mode ... 728
17.5.11.2 Master Mode ... 731
17.5.12 Error Detection ... 732
17.5.12.1 Data Consistency Check ... 732
17.5.12.2 Parity Check ... 734
17.5.12.3 Time-Out Error ... 735
17.5.12.4 Overflow Error ... 736
17.5.12.5 Overrun Error ... 738
17.5.13 Loop-Back Mode ... 741
17.5.14 CPU-Controlled High Priority Communication Function ... 742
17.5.15 Enforced Chip Select Idle Setting ... 745
17.6 Operating Procedures ... 746
17.6.1 Procedures in Direct Access Mode ... 746
17.6.1.1 Transmission/Reception in Master Mode when Job Mode is Disabled ... 746
17.6.1.2 Transmission/Reception in Master Mode when Job Mode is Enabled ... 748
17.6.2 Procedures in Transmit-Only Buffer Mode ... 750
17.6.2.1 Transmission/Reception in Master Mode when Job Mode is Disabled ... 750
17.6.2.2 Transmission/Reception in Master Mode when Job Mode is Enabled ... 752
17.6.3 Procedures in Dual Buffer Mode ... 754
17.6.3.1 Transmission/Reception in Master Mode when Job Mode is Disabled ... 754
17.6.3.2 Transmission/Reception in Master Mode when Job Mode is Enabled ... 756
17.6.3.3 Transmit/Receive in Slave Mode when Job Mode is Disabled ... 758
17.6.4 Procedures in FIFO Mode ... 760
17.6.4.1 Transmission/Reception in Master Mode when Job Mode is Disabled ... 760
17.6.4.2 Transmit/Receive Mode when Job Mode is Enabled in Master Mode ... 762
17.7 Detection and Correction of Errors in CSIHn RAM ... 764
17.7.1 ECC for the CSIHn RAM ... 764
17.7.2 Interrupt Request ... 764
17.7.3 Register Base Address ... 765
17.7.4 List of Registers ... 765
17.7.5 ECCCSIHnCTL — CSIHn ECC Control Register ... 766
17.7.6 ECCCSIHnTMC — CSIHn ECC Test Mode Control Register ... 768
17.7.7 ECCCSIHnTED — CSIHn ECC Encode/Decode Input/Output Replacement Test Register ... 770
17.7.8 ECCCSIHnTRC — CSIHn ECC Redundant Bit Data Control Test Register ... 771
17.7.9 ECCCSIHnAD0 — CSIHn ECC Error Address Register 0 ... 772
17.7.10 ECCCSIHnSYND — CSIHn ECC Decode Syndrome Data Register ... 773
17.7.11 ECCCSIHnHORD — CSIHn ECC 7-Bit Redundant Bit Data Hold Test Register ... 774
17.7.12 ECCCSIHnECRD — CSIHn ECC Encode Test Register ... 774
17.7.13 ECCCSIHnERDB — CSIHn ECC Redundant Bit Input/Output Replacement Register ... 775
17.7.14 SELB_READTEST — ECCREAD Test Select Register ... 776

Section 18 LIN Master Interface (RLIN2) ... 777

18.1 Features of RH850/F1K RLIN2 ... 777
18.1.1 Number of Units and Channels ... 777
18.1.2 Register Base Addresses ... 778
18.1.3 Clock Supply ... 778

18.1.4 Interrupt Request ... 778
18.1.5 Reset Sources ... 779
18.1.6 External Input/Output Signals ... 779
18.2 Overview ... 780
18.2.1 Functional Overview ... 780
18.2.2 Block Diagram ... 781
18.3 Registers ... 782
18.3.1 List of Registers ... 782
18.3.2 Global Registers ... 783
18.3.2.1 RLN24nGLWBR — LIN Wake-Up Baud Rate Select Register ... 783
18.3.2.2 RLN24nGLBRP0 — LIN Baud Rate Prescaler 0 Register ... 784
18.3.2.3 RLN24nGLBRP1 — LIN Baud Rate Prescaler 1 Register ... 785
18.3.2.4 RLN24nGLSTC — LIN Self-Test Control Register ... 786
18.3.3 Channel Registers ... 787
18.3.3.1 RLN24nmLiMD — LIN Mode Register ... 787
18.3.3.2 RLN24nmLiBFC — LIN Break Field Configuration Register ... 788
18.3.3.3 RLN24nmLiSC — LIN Space Configuration Register ... 789
18.3.3.4 RLN24nmLiWUP — LIN Wake-Up Configuration Register ... 790
18.3.3.5 RLN24nmLiIE — LIN Interrupt Enable Register ... 791
18.3.3.6 RLN24nmLiEDE — LIN Error Detection Enable Register ... 792
18.3.3.7 RLN24nmLiCUC — LIN Control Register ... 794
18.3.3.8 RLN24nmLiTRC — LIN Transmission Control Register ... 795
18.3.3.9 RLN24nmLiMST — LIN Mode Status Register ... 796
18.3.3.10 RLN24nmLiST — LIN Status Register ... 797
18.3.3.11 RLN24nmLiEST — LIN Error Status Register ... 799
18.3.3.12 RLN24nmLiDFC — LIN Data Field Configuration Register ... 801
18.3.3.13 RLN24nmLiIDB — LIN ID Buffer Register ... 803
18.3.3.14 RLN24nmLiCBR — LIN Checksum Buffer Register ... 804
18.3.3.15 RLN24nmLiDBRb — LIN Data Buffer b Register ... 805
18.4 Interrupt Sources ... 807
18.5 Modes ... 808
18.6 LIN Reset Mode ... 810
18.7 LIN Operation Mode ... 811
18.8 LIN Wake-Up Mode ... 811
18.9 Header Transmission/Response Transmission/Response Reception ... 812
18.9.1 Header Transmission ... 812
18.9.2 Response Transmission ... 813
18.9.3 Response Reception ... 814
18.10 Data Transmission/Reception ... 815
18.10.1 Data Transmission ... 815
18.10.2 Data Reception ... 816
18.11 Transmission/Reception Data Buffering ... 817
18.11.1 Transmission of LIN Frames ... 817
18.11.2 Reception of LIN Frames ... 818
18.12 Wake-Up Transmission/Reception ... 819

18.12.1 Wake-Up Transmission ..... 819
18.12.2 Wake-Up Reception ..... 820
18.12.3 Wake-Up Collision ..... 820
18.13 Status ..... 821
18.14 Error Status ..... 822
18.14.1 Types of Error Statuses ..... 822
18.14.2 Target Time Domain for Error Detection ..... 823
18.15 LIN Self-Test Mode ..... 824
18.15.1 Transition to LIN Self-Test Mode ..... 825
18.15.2 Transmission in LIN Self-Test Mode ..... 826
18.15.3 Reception in LIN Self-Test Mode ..... 827
18.15.4 Exiting LIN Self-Test Mode ..... 828
18.16 Baud Rate Generator ..... 829

Section 19 LIN/UART Interface (RLIN3) ..... 830
19.1 Features of RH850/F1K RLIN3 ..... 830
19.1.1 Number of Units and Channels ..... 830
19.1.2 Register Base Address ..... 831
19.1.3 Clock Supply ..... 831
19.1.4 Interrupt Request ..... 832
19.1.5 Reset Sources ..... 832
19.1.6 External Input/output Signals ..... 833
19.2 Overview ..... 834
19.2.1 Functional Overview ..... 834
19.2.2 Block Diagram ..... 838
19.2.3 Terms used in block diagram ..... 838
19.3 Registers ..... 839
19.3.1 List of Registers ..... 839
19.3.2 LIN Master Related Registers ..... 841
19.3.2.1 RLN3nLWBR — LIN Wake-Up Baud Rate Select Register ..... 841
19.3.2.2 RLN3nLBRP0 — LIN Baud Rate Prescaler 0 Register ..... 842
19.3.2.3 RLN3nLBRP1 — LIN Baud Rate Prescaler 1 Register ..... 843
19.3.2.4 RLN3nLSTC — LIN Self-Test Control Register ..... 844
19.3.2.5 RLN3nLMD — LIN Mode Register ..... 845
19.3.2.6 RLN3nLBFC — LIN Break Field Configuration Register ..... 847
19.3.2.7 RLN3nLSC — LIN Space Configuration Register ..... 848
19.3.2.8 RLN3nLWUP — LIN Wake-Up Configuration Register ..... 849
19.3.2.9 RLN3nLIE — LIN Interrupt Enable Register ..... 850
19.3.2.10 RLN3nLEDE —LIN Error Detection Enable Register ..... 852
19.3.2.11 RLN3nLCUC — LIN Control Register ..... 854
19.3.2.12 RLN3nLTRC — LIN Transmission Control Register ..... 855
19.3.2.13 RLN3nLMST — LIN Mode Status Register ..... 856
19.3.2.14 RLN3nLST — LIN Status Register ..... 857
19.3.2.15 RLN3nLEST — LIN Error Status Register ..... 859
19.3.2.16 RLN3nLDFC — LIN Data Field Configuration Register ..... 861
19.3.2.17 RLN3nLIDB — LIN ID Buffer Register ..... 863

19.3.2.18 RLN3nLCBR — LIN Checksum Buffer Register ... 864
19.3.2.19 RLN3nLDBRb — LIN Data Buffer b Register (b = 1 to 8) ... 865
19.3.3 LIN Slave Related Registers ... 867
19.3.3.1 RLN3nLWBR — LIN Wake-Up Baud Rate Select Register ... 867
19.3.3.2 RLN3nLBRP01 — LIN Baud Rate Prescaler 01 Register ... 868
19.3.3.3 RLN3nLSTC — LIN Self-Test Control Register ... 869
19.3.3.4 RLN3nLMD — LIN Mode Register ... 870
19.3.3.5 RLN3nLBFC — LIN Break Field Configuration Register ... 871
19.3.3.6 RLN3nLSC — LIN Space Configuration Register ... 872
19.3.3.7 RLN3nLWUP — LIN Wake-Up Configuration Register ... 873
19.3.3.8 RLN3nLIE — LIN Interrupt Enable Register ... 874
19.3.3.9 RLN3nLEDE — LIN Error Detection Enable Register ... 876
19.3.3.10 RLN3nLCUC — LIN Control Register ... 878
19.3.3.11 RLN3nLTRC — LIN Transmission Control Register ... 879
19.3.3.12 RLN3nLMST — LIN Mode Status Register ... 880
19.3.3.13 RLN3nLST — LIN Status Register ... 881
19.3.3.14 RLN3nLEST — LIN Error Status Register ... 883
19.3.3.15 RLN3nLDFC — LIN Data Field Configuration Register ... 885
19.3.3.16 RLN3nLIDB — LIN ID Buffer Register ... 887
19.3.3.17 RLN3nLCBR — LIN Checksum Buffer Register ... 888
19.3.3.18 RLN3nLDBRb — LIN Data Buffer b Register (b = 1 to 8) ... 889
19.3.4 UART Related Registers ... 890
19.3.4.1 RLN3nLWBR — LIN Wake-Up Baud Rate Select Register ... 890
19.3.4.2 RLN3nLBRP01 — UART Baud Rate Prescaler 01 Register ... 891
19.3.4.3 RLN3nLMD — UART Mode Register ... 892
19.3.4.4 RLN3nLBFC — UART Configuration Register ... 893
19.3.4.5 RLN3nLSC — UART Space Configuration Register ... 895
19.3.4.6 RLN3nLEDE —UART Error Detection Enable Register ... 896
19.3.4.7 RLN3nLCUC — UART Control Register ... 897
19.3.4.8 RLN3nLTRC — UART Transmission Control Register ... 898
19.3.4.9 RLN3nLMST — UART Mode Status Register ... 899
19.3.4.10 RLN3nLST — UART Status Register ... 900
19.3.4.11 RLN3nLEST — UART Error Status Register ... 902
19.3.4.12 RLN3nLDFC — UART Data Field Configuration Register ... 904
19.3.4.13 RLN3nLIDB — UART ID Buffer Register ... 905
19.3.4.14 RLN3nLUDB0 — UART Data Buffer 0 Register ... 905
19.3.4.15 RLN3nLDBRb — UART Data Buffer b Register (b = 1 to 8) ... 906
19.3.4.16 RLN3nLUOER — UART Operation Enable Register ... 907
19.3.4.17 RLN3nLUOR1 — UART Option Register 1 ... 908
19.3.4.18 RLN3nLUTDR — UART Transmission Data Register ... 910
19.3.4.19 RLN3nLURDR — UART Reception Data Register ... 911
19.3.4.20 RLN3nLUWTDR — UART Wait Transmission Data Register ... 912
19.4 Interrupt Sources ... 913
19.5 Modes ... 914
19.6 LIN Reset Mode ... 916
19.7 LIN Mode ... 917
19.7.1 LIN Master Mode ... 920
19.7.1.1 Header Transmission ... 920

19.7.1.2 Response Transmission ... 921
19.7.1.3 Response Reception ... 922
19.7.2 LIN Slave Mode ... 923
19.7.2.1 Header Reception ... 923
19.7.2.2 Response Transmission ... 926
19.7.2.3 Response Reception ... 927
19.7.2.4 No-response Request ... 928
19.7.3 Data Transmission/Reception ... 929
19.7.3.1 Data Transmission ... 929
19.7.3.2 Data Reception ... 930
19.7.4 Transmission/Reception Data Buffering ... 931
19.7.4.1 Transmission of LIN Frames ... 931
19.7.4.2 Reception of LIN Frames ... 932
19.7.4.3 Multi-Byte Response Transmission/Reception Function ... 933
19.7.5 Wake-up Transmission/Reception ... 934
19.7.5.1 Wake-up Transmission ... 934
19.7.5.2 Wake-up Reception ... 935
19.7.5.3 Wakeup Collision ... 935
19.7.6 Statuses ... 936
19.7.7 Error Statuses ... 938
19.7.7.1 LIN Master Mode ... 938
19.7.7.2 LIN Slave Mode ... 940
19.8 UART Mode ... 942
19.8.1 Transmission ... 942
19.8.1.1 Continuous Transmission ... 944
19.8.1.2 UART Buffer Transmission ... 945
19.8.1.3 Data Transmission ... 947
19.8.1.4 Transmission Start Wait Function ... 948
19.8.2 Reception ... 949
19.8.2.1 Data Reception ... 950
19.8.3 Expansion Bits ... 951
19.8.3.1 Expansion Bit Transmission ... 951
19.8.3.2 Expansion Bit Reception ... 951
19.8.3.3 Expansion Bit Reception (with Expansion Bit Comparison) ... 952
19.8.3.4 Expansion Bit Reception (with Data Comparison) ... 953
19.8.4 Statuses ... 954
19.8.5 Error Statuses ... 955
19.9 LIN Self-Test Mode ... 956
19.9.1 Transition to LIN Self-Test Mode ... 958
19.9.2 Transmission in LIN Master Self-Test Mode ... 959
19.9.3 Reception in LIN Master Self-Test Mode ... 960
19.9.4 Transmission in LIN Slave Self-Test Mode ... 961
19.9.5 Reception in LIN Slave Self-Test Mode ... 963
19.9.6 Exiting LIN Self-Test Mode ... 964
19.10 Baud Rate Generator ... 965
19.10.1 LIN Master Mode ... 965
19.10.2 LIN Slave Mode ... 966

19.10.3 UART Mode ... 967
19.11 Noise Filter ... 968

Section 20 I2C Bus Interface (RIIC) ... 970
20.1 Features of RH850/F1K RIIC ... 970
20.1.1 Number of Units and Channels ... 970
20.1.2 Register Base Address ... 970
20.1.3 Clock Supply ... 970
20.1.4 Interrupt Requests ... 971
20.1.5 Reset Sources ... 971
20.1.6 External Input/Output Signals ... 971
20.2 Overview ... 972
20.2.1 Functional Overview ... 972
20.2.2 Block Diagram ... 974
20.3 Registers ... 976
20.3.1 List of Registers ... 976
20.3.2 RIICnCR1 — I2C Bus Control Register 1 ... 977
20.3.3 RIICnCR2 — I2C Bus Control Register 2 ... 980
20.3.4 RIICnMR1 — I2C Bus Mode Register 1 ... 984
20.3.5 RIICnMR2 — I2C Bus Mode Register 2 ... 986
20.3.6 RIICnMR3 — I2C Bus Mode Register 3 ... 988
20.3.7 RIICnFER — I2C Bus Function Enable Register ... 991
20.3.8 RIICnSER — I2C Bus Status Enable Register ... 993
20.3.9 RIICnIER — I2C Bus Interrupt Enable Register ... 995
20.3.10 RIICnSR1 — I2C Bus Status Register 1 ... 997
20.3.11 RIICnSR2 — I2C Bus Status Register 2 ... 1000
20.3.12 RIICnSARy — I2C Slave Address Register y (y = 0 to 2) ... 1005
20.3.13 RIICnBRL — I2C Bus Bit Rate Low-Level Register ... 1007
20.3.14 RIICnBRH — I2C Bus Bit Rate High-Level Register ... 1008
20.3.15 RIICnDRT — I2C Bus Transmit Data Register ... 1011
20.3.16 RIICnDRR — I2C Bus Receive Data Register ... 1012
20.3.17 RIICnDRS — I2C Bus Shift Register ... 1013
20.4 Interrupt Sources ... 1014
20.5 Operation ... 1015
20.5.1 Communication Data Format ... 1015
20.5.2 Initial Settings ... 1016
20.5.3 Master Transmit Operation ... 1017
20.5.4 Master Receive Operation ... 1021
20.5.5 Slave Transmit Operation ... 1027
20.5.6 Slave Receive Operation ... 1030
20.6 SCL Synchronization Circuit ... 1033
20.7 Facility for Delaying SDA Output ... 1034
20.8 Digital Noise-Filter Circuits ... 1035
20.9 Address Match Detection ... 1036

20.9.1 Slave-Address Match Detection ..... 1036
20.9.2 Detection of the General Call Address ..... 1038
20.9.3 Device-ID Address Detection ..... 1039
20.10 Automatic Low-Hold Function for SCL ..... 1041
20.10.1 Function to Prevent Wrong Transmission of Transmit Data ..... 1041
20.10.2 NACK Reception Transfer Suspension Function ..... 1042
20.10.3 Function to Prevent Failure to Receive Data ..... 1043
20.11 Arbitration-Lost Detection Functions ..... 1045
20.11.1 Master Arbitration-Lost Detection (MALE Bit) ..... 1045
20.11.2 Function to Detect Loss of Arbitration during NACK Transmission (NALE Bit) ..... 1047
20.11.3 Slave Arbitration-Lost Detection (SALE Bit) ..... 1048
20.12 Start Condition/Restart Condition/Stop Condition Issuing Function ..... 1049
20.12.1 Issuing a Start Condition ..... 1049
20.12.2 Issuing a Restart Condition ..... 1049
20.12.3 Issuing a Stop Condition ..... 1051
20.13 Bus Hanging ..... 1052
20.13.1 Timeout Function ..... 1052
20.13.2 Extra SCL Clock Cycle Output Function ..... 1054
20.13.3 RIIC Reset and Internal Reset ..... 1055
20.14 Reset Function of RIIC ..... 1056

Section 21 CAN Interface (RS-CAN) ..... 1058
21.1 Features of RH850/F1K RS-CANFD ..... 1058
21.1.1 Number of Units and Channels ..... 1058
21.2 RS-CANFD ..... 1059
21.2.1 Register Base Address ..... 1060
21.2.2 Clock Supply ..... 1060
21.2.3 Interrupt Requests ..... 1061
21.2.4 Reset Factors ..... 1063
21.2.5 External Input/Output Signals ..... 1063
21.3 Overview ..... 1064
21.3.1 Functional Overview ..... 1064
21.3.2 Interface Modes ..... 1066
21.3.3 Block Diagram ..... 1067
21.4 Registers (Classical CAN Mode) ..... 1068
21.4.1 List of Registers ..... 1068
21.4.2 Details of Channel-Related Registers ..... 1072
21.4.2.1 RSCANnCmCFG — Channel Configuration Register (m = 0 to 5) ..... 1072
21.4.2.2 RSCANnCmCTR — Channel Control Register (m = 0 to 5) ..... 1074
21.4.2.3 RSCANnCmSTS — Channel Status Register (m = 0 to 5) ..... 1079
21.4.2.4 RSCANnCmERFL — Channel Error Flag Register (m = 0 to 5) ..... 1082
21.4.3 Details of Global-Related Registers ..... 1086
21.4.3.1 RSCANnGCFG — Global Configuration Register ..... 1086
21.4.3.2 RSCANnGCTR — Global Control Register ..... 1089
21.4.3.3 RSCANnGSTS — Global Status Register ..... 1091

21.4.3.4 RSCANnGERFL — Global Error Flag Register ... 1093
21.4.3.5 RSCANnGTSC — Global Timestamp Counter Register ... 1095
21.4.3.6 RSCANnGTINTSTS0 — Global TX Interrupt Status Register 0 ... 1096
21.4.3.7 RSCANnGTINTSTS1 — Global TX Interrupt Status Register 1 ... 1099
21.4.3.8 RSCANnGFDCFG — Global FD Configuration Register ... 1101
21.4.4 Details of Receive Rule-related Registers ... 1102
21.4.4.1 RSCANnGAFLECTR — Receive Rule Entry Control Register ... 1102
21.4.4.2 RSCANnGAFLCFG0 — Receive Rule Configuration Register 0 ... 1103
21.4.4.3 RSCANnGAFLCFG1 — Receive Rule Configuration Register 1 ... 1105
21.4.4.4 RSCANnGAFLIDj — Receive Rule ID Register (j = 0 to 15) ... 1106
21.4.4.5 RSCANnGAFLMj — Receive Rule Mask Register (j = 0 to 15) ... 1108
21.4.4.6 RSCANnGAFLP0j — Receive Rule Pointer 0 Register (j = 0 to 15) ... 1109
21.4.4.7 RSCANnGAFLP1j — Receive Rule Pointer 1 Register (j = 0 to 15) ... 1111
21.4.5 Details of Receive Buffer-Related Registers ... 1112
21.4.5.1 RSCANnRMNB — Receive Buffer Number Register ... 1112
21.4.5.2 RSCANnRMNDy — Receive Buffer New Data Register (y = 0 to 2) ... 1113
21.4.5.3 RSCANnRMIDq — Receive Buffer ID Register (q = 0 to 95) ... 1114
21.4.5.4 RSCANnRMPTRq — Receive Buffer Pointer Register (q = 0 to 95) ... 1115
21.4.5.5 RSCANnRMDF0q — Receive Buffer Data Field 0 Register (q = 0 to 95) ... 1116
21.4.5.6 RSCANnRMDF1q — Receive Buffer Data Field 1 Register (q = 0 to 95) ... 1117
21.4.6 Details of Receive FIFO Buffer-Related Registers ... 1118
21.4.6.1 RSCANnRFCCx — Receive FIFO Buffer Configuration and Control Register (x = 0 to 7) ... 1118
21.4.6.2 RSCANnRFSTSx — Receive FIFO Buffer Status Register (x = 0 to 7) ... 1120
21.4.6.3 RSCANnRFPCTRx — Receive FIFO Buffer Pointer Control Register (x = 0 to 7) ... 1122
21.4.6.4 RSCANnRFIDx — Receive FIFO Buffer Access ID Register (x = 0 to 7) ... 1123
21.4.6.5 RSCANnRFPTRx — Receive FIFO Buffer Access Pointer Register (x = 0 to 7) 1124
21.4.6.6 RSCANnRFDF0x — Receive FIFO Buffer Access Data Field 0 Register (x = 0 to 7) ... 1125
21.4.6.7 RSCANnRFDF1x — Receive FIFO Buffer Access Data Field 1 Register (x = 0 to 7) ... 1126
21.4.7 Details of Transmit/Receive FIFO Buffer-Related Registers ... 1127
21.4.7.1 RSCANnCFCCk — Transmit/receive FIFO Buffer Configuration and Control Register k (k = 0 to 17) ... 1127
21.4.7.2 RSCANnCFSTSk — Transmit/receive FIFO Buffer Status Register (k = 0 to 17) ... 1131
21.4.7.3 RSCANnCFPCTRk — Transmit/receive FIFO Buffer Pointer Control Register (k = 0 to 17) ... 1134
21.4.7.4 RSCANnCFIDk — Transmit/receive FIFO Buffer Access ID Register (k = 0 to 17) ... 1136
21.4.7.5 RSCANnCFPTRk — Transmit/receive FIFO Buffer Access Pointer Register (k = 0 to 17) ... 1138
21.4.7.6 RSCANnCFDF0k — Transmit/receive FIFO Buffer Access Data Field 0 Register (k = 0 to 17) ... 1140
21.4.7.7 RSCANnCFDF1k — Transmit/receive FIFO Buffer Access Data Field 1 Register (k = 0 to 17) ... 1141
21.4.8 Details of FIFO Status-Related Registers ... 1142
21.4.8.1 RSCANnFESTS — FIFO Empty Status Register ... 1142
21.4.8.2 RSCANnFFSTS — FIFO Full Status Register ... 1144
21.4.8.3 RSCANnFMSTS — FIFO Message Lost Status Register ... 1146

21.4.8.4 RSCANnRFISTS — Receive FIFO Buffer Interrupt Flag Status Register ..... 1148
21.4.8.5 RSCANnCFRISTS — Transmit/receive FIFO Buffer Receive Interrupt Flag Status Register ..... 1149
21.4.8.6 RSCANnCFTISTS — Transmit/receive FIFO Buffer Transmit Interrupt Flag Status Register ..... 1151
21.4.9 Details of Transmit Buffer-Related Registers ..... 1153
21.4.9.1 RSCANnTMCp — Transmit Buffer Control Register (p = 0 to 95) ..... 1153
21.4.9.2 RSCANnTMSTSp — Transmit Buffer Status Register (p = 0 to 95) ..... 1155
21.4.9.3 RSCANnTMIDp — Transmit Buffer ID Register (p = 0 to 95) ..... 1157
21.4.9.4 RSCANnTMPTRp — Transmit Buffer Pointer Register (p = 0 to 95) ..... 1159
21.4.9.5 RSCANnTMDF0p — Transmit Buffer Data Field 0 Register (p = 0 to 95) ..... 1161
21.4.9.6 RSCANnTMDF1p — Transmit Buffer Data Field 1 Register (p = 0 to 95) ..... 1162
21.4.9.7 RSCANnTMIECy — Transmit Buffer Interrupt Enable Configuration Register (y = 0 to 2) ..... 1163
21.4.10 Details of Transmit Buffer Status-Related Registers ..... 1165
21.4.10.1 RSCANnTMTRSTSy — Transmit Buffer Transmit Request Status Register (y = 0 to 2) ..... 1165
21.4.10.2 RSCANnTMTARSTSy — Transmit Buffer Transmit Abort Request Status Register (y = 0 to 2) ..... 1167
21.4.10.3 RSCANnTMTCSTSy — Transmit Buffer Transmit Complete Status Register (y = 0 to 2) ..... 1169
21.4.10.4 RSCANnTMTASTSy — Transmit Buffer Transmit Abort Status Register (y = 0 to 2) ..... 1171
21.4.11 Details of Transmit Queue-Related Registers ..... 1173
21.4.11.1 RSCANnTXQCCm — Transmit Queue Configuration and Control Register (m = 0 to 5) ..... 1173
21.4.11.2 RSCANnTXQSTSm — Transmit Queue Status Register (m = 0 to 5) ..... 1175
21.4.11.3 RSCANnTXQPCTRm — Transmit Queue Pointer Control Register (m = 0 to 5) ..... 1177
21.4.12 Details of Transmit history-related Registers ..... 1178
21.4.12.1 RSCANnTHLCCm — Transmit History Configuration and Control Register (m = 0 to 5) ..... 1178
21.4.12.2 RSCANnTHLSTSm — Transmit History Status Register (m = 0 to 5) ..... 1180
21.4.12.3 RSCANnTHLPCTRm — Transmit History Pointer Control Register (m = 0 to 5) 1182
21.4.12.4 RSCANnTHLACCm — Transmit History Access Register (m = 0 to 5) ..... 1183
21.4.13 Details of Test-Related Registers ..... 1185
21.4.13.1 RSCANnGTSTCFG — Global Test Configuration Register ..... 1185
21.4.13.2 RSCANnGTSTCTR — Global Test Control Register ..... 1187
21.4.13.3 RSCANnGLOCKK — Global Lock Key Register ..... 1188
21.4.13.4 RSCANnRPGACCr — RAM Test Page Access Register (r = 0 to 63) ..... 1189
21.5 Registers (CAN FD Mode) ..... 1190
21.5.1 List of Registers ..... 1190
21.5.2 Details of Interface Mode-Related Registers ..... 1195
21.5.2.1 RSCFDnCFDGRMCFG — Global Interface Mode Select Register ..... 1195
21.5.3 Details of Channel-Related Registers ..... 1196
21.5.3.1 RSCFDnCFDCmNCFG — Channel Nominal Bit Rate Configuration Register (m = 0 to 5) ..... 1196
21.5.3.2 RSCFDnCFDCmCTR — Channel Control Register (m = 0 to 5) ..... 1198
21.5.3.3 RSCFDnCFDCmSTS — Channel Status Register (m = 0 to 5) ..... 1203
21.5.3.4 RSCFDnCFDCmERFL — Channel Error Flag Register (m = 0 to 5) ..... 1206

21.5.3.5 RSCFDnCFDCmDCFG — Channel Data Bit Rate Configuration register (m = 0 to 5) ... 1210
21.5.3.6 RSCFDnCFDCmFDCFG — Channel CAN FD Configuration Register (m = 0 to 5) ... 1213
21.5.3.7 RSCFDnCFDCmFDCTR — Channel CAN FD Control Register (m = 0 to 5) ... 1217
21.5.3.8 RSCFDnCFDCmFDSTS — Channel CAN FD Status Register (m = 0 to 5) ... 1218
21.5.3.9 RSCFDnCFDCmFDCRC — Channel CAN FD CRC Register (m = 0 to 5) ... 1220
21.5.4 Details of Global-Related Registers ... 1222
21.5.4.1 RSCFDnCFDGCFG — Global Configuration Register ... 1222
21.5.4.2 RSCFDnCFDGCTR — Global Control Register ... 1226
21.5.4.3 RSCFDnCFDGSTS — Global Status Register ... 1228
21.5.4.4 RSCFDnCFDGERFL — Global Error Flag Register ... 1230
21.5.4.5 RSCFDnCFDGTSC — Global Timestamp Counter Register ... 1232
21.5.4.6 RSCFDnCFDGTINTSTS0 — Global TX Interrupt Status Register 0 ... 1233
21.5.4.7 RSCFDnCFDGTINTSTS1 — Global TX Interrupt Status Register 1 ... 1236
21.5.4.8 RSCFDnCFDGFDCFG — Global FD Configuration Register ... 1238
21.5.5 Details of Receive Rule-related Registers ... 1239
21.5.5.1 RSCFDnCFDGAFLECTR — Receive Rule Entry Control Register ... 1239
21.5.5.2 RSCFDnCFDGAFLCFG0 — Receive Rule Configuration Register 0 ... 1240
21.5.5.3 RSCFDnCFDGAFLCFG1 — Receive Rule Configuration Register 1 ... 1242
21.5.5.4 RSCFDnCFDGAFLIDj — Receive Rule ID Register (j = 0 to 15) ... 1243
21.5.5.5 RSCFDnCFDGAFLMj — Receive Rule Mask Register (j = 0 to 15) ... 1245
21.5.5.6 RSCFDnCFDGAFLP0_j — Receive Rule Pointer 0 Register (j = 0 to 15) ... 1246
21.5.5.7 RSCFDnCFDGAFLP1_j — Receive Rule Pointer 1 Register (j = 0 to 15) ... 1248
21.5.6 Details of Receive Buffer-related Registers ... 1249
21.5.6.1 RSCFDnCFDRMNB — Receive Buffer Number Register ... 1249
21.5.6.2 RSCFDnCFDRMNDy — Receive Buffer New Data Register (y = 0 to 2) ... 1250
21.5.6.3 RSCFDnCFDRMIDq — Receive Buffer ID Register (q = 0 to 95) ... 1251
21.5.6.4 RSCFDnCFDRMPTRq — Receive Buffer Pointer Register (q = 0 to 95) ... 1252
21.5.6.5 RSCFDnCFDRMFDSTSq — Receive Buffer CAN FD Status Register (q = 0 to 95) ... 1254
21.5.6.6 RSCFDnCFDRMDFb_q — Receive Buffer Data Field b Register (b = 0 to 4, q = 0 to 95) ... 1255
21.5.7 Details of Receive FIFO Buffer-related Registers ... 1256
21.5.7.1 RSCFDnCFDRFCCx — Receive FIFO Buffer Configuration and Control Register (x = 0 to 7) ... 1256
21.5.7.2 RSCFDnCFDRFSTSx — Receive FIFO Buffer Status Register (x = 0 to 7) ... 1258
21.5.7.3 RSCFDnCFDRFPCTRx — Receive FIFO Buffer Pointer Control Register (x = 0 to 7) ... 1260
21.5.7.4 RSCFDnCFDRFIDx — Receive FIFO Buffer Access ID Register (x = 0 to 7) ... 1261
21.5.7.5 RSCFDnCFDRFPTRx — Receive FIFO Buffer Access Pointer Register (x = 0 to 7) ... 1262
21.5.7.6 RSCFDnCFDRFFDSTSx — Receive FIFO CAN FD Status Register (x = 0 to 7) ... 1264
21.5.7.7 RSCFDnCFDRFDFd_x — Receive FIFO Buffer Access Data Field d Register (d = 0 to 15, x = 0 to 7) ... 1265
21.5.8 Transmit/Receive FIFO Buffer Related Registers ... 1266
21.5.8.1 RSCFDnCFDCFCCk — Transmit/receive FIFO Buffer Configuration and Control Register k (k = 0 to 17) ... 1266
21.5.8.2 RSCFDnCFDCFSTSk — Transmit/receive FIFO Buffer Status Register (k = 0 to 17) ... 1270

21.5.8.3 RSCFDnCFDCFPCTRk — Transmit/receive FIFO Buffer Pointer Control Register (k = 0 to 17) ... 1274
21.5.8.4 RSCFDnCFDCFIDk — Transmit/receive FIFO Buffer Access ID Register (k = 0 to 17) ... 1276
21.5.8.5 RSCFDnCFDCFPTRk — Transmit/receive FIFO Buffer Access Pointer Register (k = 0 to 17) ... 1278
21.5.8.6 RSCFDnCFDCFFDCSTSk — Transmit/Receive FIFO CAN FD Configuration/ Status Register (k = 0 to 17) ... 1280
21.5.8.7 RSCFDnCFDCFDFd_k — Transmit/receive FIFO Buffer Access Data Field d Register (d = 0 to 15, k = 0 to 17) ... 1282
21.5.9 Details of FIFO Status-related Registers ... 1283
21.5.9.1 RSCFDnCFDFESTS — FIFO Empty Status Register ... 1283
21.5.9.2 RSCFDnCFDFFSTS — FIFO Full Status Register ... 1285
21.5.9.3 RSCFDnCFDFMSTS — FIFO Message Lost Status Register ... 1287
21.5.9.4 RSCFDnCFDRFISTS — Receive FIFO Buffer Interrupt Flag Status Register ... 1289
21.5.9.5 RSCFDnCFDCFRISTS — Transmit/receive FIFO Buffer Receive Interrupt Flag Status Register ... 1290
21.5.9.6 RSCFDnCFDCFTISTS — Transmit/receive FIFO Buffer Transmit Interrupt Flag Status Register ... 1292
21.5.10 Details of FIFO DMA-Related Registers ... 1294
21.5.10.1 RSCFDnCFDCDTCT — DMA Enable Register ... 1294
21.5.10.2 RSCFDnCFDCDTSTS — DMA Status Register ... 1296
21.5.11 Details of Transmit Buffer-related Registers ... 1298
21.5.11.1 RSCFDnCFDTMCp — Transmit Buffer Control Register (p = 0 to 95) ... 1298
21.5.11.2 RSCFDnCFDTMSTSp — Transmit Buffer Status Register (p = 0 to 95) ... 1300
21.5.11.3 RSCFDnCFDTMIDp — Transmit Buffer ID Register (p = 0 to 95) ... 1302
21.5.11.4 RSCFDnCFDTMPTRp — Transmit Buffer Pointer Register (p = 0 to 95) ... 1304
21.5.11.5 RSCFDnCFDTMFDCTRp — Transmit Buffer CAN FD Configuration Register (p = 0 to 95) ... 1306
21.5.11.6 RSCFDnCFDTMDFb_p — Transmit Buffer Data Field b Register (b = 0 to 4, p = 0 to 95) ... 1308
21.5.11.7 RSCFDnCFDTMIECy — Transmit Buffer Interrupt Enable Configuration Register (y = 0 to 2) ... 1309
21.5.12 Details of Transmit Buffer Status-related Registers ... 1311
21.5.12.1 RSCFDnCFDTMTRSTSy — Transmit Buffer Transmit Request Status Register (y = 0 to 2) ... 1311
21.5.12.2 RSCFDnCFDTMTARSTSy — Transmit Buffer Transmit Abort Request Status Register (y = 0 to 2) ... 1313
21.5.12.3 RSCFDnCFDTMTCSTSy — Transmit Buffer Transmit Complete Status Register (y = 0 to 2) ... 1315
21.5.12.4 RSCFDnCFDTMTASTSy — Transmit Buffer Transmit Abort Status Register (y = 0 to 2) ... 1317
21.5.13 Details of Transmit Queue-related Registers ... 1319
21.5.13.1 RSCFDnCFDTXQCCm — Transmit Queue Configuration and Control Register (m = 0 to 5) ... 1319
21.5.13.2 RSCFDnCFDTXQSTSm — Transmit Queue Status Register (m = 0 to 5) ... 1321
21.5.13.3 RSCFDnCFDTXQPCTRm — Transmit Queue Pointer Control Register (m = 0 to 5) ... 1323
21.5.14 Details of Transmit History-related Registers ... 1324
21.5.14.1 RSCFDnCFDTHLCCm — Transmit History Configuration and Control Register (m = 0 to 5) ... 1324
21.5.14.2 RSCFDnCFDTHLSTSm — Transmit History Status Register (m = 0 to 5) ... 1326

21.5.14.3 RSCFDnCFDTHLPCTRm — Transmit History Pointer Control Register (m = 0 to 5) ... 1328
21.5.14.4 RSCFDnCFDTHLACCm — Transmit History Access Register (m = 0 to 5) ... 1329
21.5.15 Details of Test-related Registers ... 1331
21.5.15.1 RSCFDnCFDGTSTCFG — Global Test Configuration Register ... 1331
21.5.15.2 RSCFDnCFDGTSTCTR — Global Test Control Register ... 1333
21.5.15.3 RSCFDnCFDGLOCKK — Global Lock Key Register ... 1334
21.5.15.4 RSCFDnCFDRPGACCr — RAM Test Page Access Register (r = 0 to 63) ... 1335
21.6 Interrupt Sources and DMA Trigger ... 1336
21.6.1 Interrupt Sources ... 1336
21.6.2 DMA Trigger (Only in CAN FD Mode) ... 1340
21.7 CAN Modes ... 1341
21.7.1 Global Modes ... 1341
21.7.1.1 Global Stop Mode ... 1343
21.7.1.2 Global Reset Mode ... 1343
21.7.1.3 Global Test Mode ... 1343
21.7.1.4 Global Operating Mode ... 1343
21.7.2 Channel Modes ... 1344
21.7.2.1 Channel Stop Mode ... 1345
21.7.2.2 Channel Reset Mode ... 1345
21.7.2.3 Channel Halt Mode ... 1346
21.7.2.4 Channel Communication Mode ... 1346
21.7.2.5 Bus Off State ... 1347
21.7.3 Initializing Registers by Transition to CAN Mode ... 1348
21.8 Reception Function ... 1350
21.8.1 Data Processing Using the Receive Rule Table ... 1350
21.8.1.1 Acceptance Filter Processing ... 1351
21.8.1.2 DLC Filter Processing ... 1352
21.8.1.3 Routing Processing ... 1352
21.8.1.4 Label Addition Processing ... 1352
21.8.1.5 Mirror Function Processing ... 1352
21.8.1.6 Timestamp ... 1353
21.9 Transmission Functions ... 1354
21.9.1 Transmit Priority Determination ... 1355
21.9.2 Transmission Using Transmit Buffers ... 1355
21.9.2.1 Transmit Abort Function ... 1355
21.9.2.2 One-Shot Transmission Function (Retransmission Disabling Function) ... 1356
21.9.2.3 Transmit Buffer Merge Mode (Only in CAN FD Mode) ... 1356
21.9.3 Transmission Using FIFO Buffers ... 1356
21.9.3.1 Interval Transmission Function ... 1357
21.9.4 Transmission Using Transmit Queues ... 1360
21.9.5 Transmit Data Padding (Only in CAN FD Mode) ... 1360
21.9.6 Transmit History Function ... 1360
21.10 Test Function ... 1362
21.10.1 Standard Test Mode ... 1362
21.10.2 Listen-Only Mode ... 1362
21.10.3 Self-Test Mode (Loopback Mode) ... 1363

21.10.3.1 Self-Test Mode 0 (External Loopback Mode) ... 1363
21.10.3.2 Self-Test Mode 1 (Internal Loopback Mode) ... 1364
21.10.4 Restricted Operation Mode (Only in CAN FD Mode) ... 1364
21.10.5 RAM Test ... 1364
21.10.6 Inter-Channel Communication Test ... 1365
21.10.6.1 CRC Error Test ... 1366
21.11 RS-CANFD Setting Procedure ... 1367
21.11.1 Initial Settings ... 1367
21.11.1.1 Clock Setting ... 1369
21.11.1.2 Bit Timing Setting ... 1369
21.11.1.3 Communication Speed Setting ... 1371
21.11.1.4 Receive Rule Setting ... 1373
21.11.1.5 Buffer Setting ... 1374
21.11.1.6 Transmitter Delay Compensation (Only in CAN FD Mode) ... 1376
21.11.2 Reception Procedure ... 1377
21.11.2.1 Receive Buffer Reading Procedure ... 1377
21.11.2.2 FIFO Buffer Reading Procedure ... 1379
21.11.2.3 FIFO Buffer Reading Procedure by DMA Transfer ... 1383
21.11.3 Transmission Procedure ... 1384
21.11.3.1 Procedure for Transmission from Transmit Buffers ... 1384
21.11.3.2 Procedure for Transmission from Transmit/Receive FIFO Buffers ... 1389
21.11.3.3 Procedure for Transmission from the Transmit Queue ... 1393
21.11.3.4 Transmit History Buffer Reading Procedure ... 1394
21.11.4 Test Settings ... 1395
21.11.4.1 Self-Test Mode Setting Procedure ... 1395
21.11.4.2 Procedure for Releasing the Protection ... 1396
21.11.4.3 RAM Test Setting Procedure ... 1397
21.11.4.4 Inter-Channel Communication Test Setting Procedure ... 1398
21.12 Detection and Correction of Errors in RS-CAN0 RAM ... 1400
21.12.1 ECC for the RSCAN0 RAM ... 1400
21.12.2 Interrupt Request ... 1400
21.12.3 List of ECC Registers ... 1401
21.12.4 ECCRCAN0CTL_PHY1/ECCRCANFD0CTL_PHY2 — RSCANn ECC Control Register ... 1402
21.12.5 ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2 — RSCANn ECC Test Mode Control Register ... 1404
21.12.6 ECCRCAN0TED_PHY1/ECCRCANFD0TED_PHY2 — RSCAN0 ECC Encode/ Decode Input/Output Replacement Test Register ... 1406
21.12.7 ECCRCAN0TRC_PHY1/ECCRCANFD0TRC_PHY2 — RSCAN0 ECC Redundant Bit Data Control Test Register ... 1407
21.12.8 ECCRCAN0AD0_PHY1/ECCRCANFD0AD0_PHY2 — RSCAN0 ECC Error Address Register 0 ... 1408
21.12.9 ECCRCAN0SYND_PHY1/ECCRCANFD0SYND_PHY2 — RSCAN0 ECC Decode Syndrome Data Register ... 1409
21.12.10 ECCRCAN0HORD_PHY1/ECCRCANFD0HORD_PHY2 — RSCAN0 ECC 7-Bit Redundant Bit Data Hold Test Register ... 1410
21.12.11 ECCRCAN0ECRD_PHY1/ECCRCANFD0ECRD_PHY2 — RSCAN0 ECC Encode Test Register ... 1411

21.12.12 ECCRCAN0ERDB_PHY1/ECCRCANFD0ERDB_PHY2 — RSCAN0 ECC Redundant Bit Input/Output Replacement Register ... 1412
21.12.13 SELB_READTEST — ECCREAD Test Select Register ... 1412
21.13 Notes on the RS-CANFD Module ... 1413
21.14 RS-CAN ... 1415
21.14.1 Register Base Address ... 1416
21.14.2 Clock Supply ... 1416
21.14.3 Interrupt Request ... 1417
21.14.4 Reset Sources ... 1417
21.14.5 External Input/Output Signals ... 1417
21.15 Overview ... 1418
21.15.1 Functional Overview ... 1418
21.15.2 Block Diagram ... 1420
21.16 Registers ... 1421
21.16.1 List of Registers ... 1421
21.16.2 RSCAN1CmCFG — Channel Configuration Register (m = 6) ... 1432
21.16.3 RSCAN1CmCTR — Channel Control Register (m = 6) ... 1434
21.16.4 RSCAN1CmSTS — Channel Status Register (m = 6) ... 1438
21.16.5 RSCAN1CmERFL — Channel Error Flag Register (m = 6) ... 1440
21.16.6 RSCAN1GCFG — Global Configuration Register ... 1444
21.16.7 RSCAN1GCTR — Global Control Register ... 1447
21.16.8 RSCAN1GSTS — Global Status Register ... 1449
21.16.9 RSCAN1GERFL — Global Error Flag Register ... 1451
21.16.10 RSCAN1GTINTSTS0 — Global TX Interrupt Status Register 0 ... 1453
21.16.11 RSCAN1GTSC — Global Timestamp Counter Register ... 1455
21.16.12 RSCAN1GAFLECTR — Receive Rule Entry Control Register ... 1456
21.16.13 RSCAN1GAFLCFG0 — Receive Rule Configuration Register 0 ... 1457
21.16.14 RSCAN1GAFLIDj — Receive Rule ID Register (j = 0 to 15) ... 1458
21.16.15 RSCAN1GAFLMj — Receive Rule Mask Register (j = 0 to 15) ... 1460
21.16.16 RSCAN1GAFLP0j — Receive Rule Pointer 0 Register (j = 0 to 15) ... 1461
21.16.17 RSCAN1GAFLP1j — Receive Rule Pointer 1 Register (j = 0 to 15) ... 1463
21.16.18 RSCAN1RMNB — Receive Buffer Number Register ... 1464
21.16.19 RSCAN1RMNDy — Receive Buffer New Data Register (y = 0) ... 1465
21.16.20 RSCAN1RMIDq — Receive Buffer ID Register (q = 0 to 15) ... 1466
21.16.21 RSCAN1RMPTRq — Receive Buffer Pointer Register (q = 0 to 15) ... 1467
21.16.22 RSCAN1RMDF0q — Receive Buffer Data Field 0 Register (q = 0 to 15) ... 1468
21.16.23 RSCAN1RMDF1q — Receive Buffer Data Field 1 Register (q = 0 to 15) ... 1469
21.16.24 RSCAN1RFCCx — Receive FIFO Buffer Configuration and Control Register (x = 0 to 7) ... 1470
21.16.25 RSCAN1RFSTSx — Receive FIFO Buffer Status Register (x = 0 to 7) ... 1472
21.16.26 RSCAN1RFPCTRx — Receive FIFO Buffer Pointer Control Register (x = 0 to 7) ... 1474
21.16.27 RSCAN1RFIDx — Receive FIFO Buffer Access ID Register (x = 0 to 7) ... 1475
21.16.28 RSCAN1RFPTRx — Receive FIFO Buffer Access Pointer Register (x = 0 to 7) ... 1476
21.16.29 RSCAN1RFDF0x — Receive FIFO Buffer Access Data Field 0 Register (x = 0 to 7) ... 1477
21.16.30 RSCAN1RFDF1x — Receive FIFO Buffer Access Data Field 1 Register (x = 0 to 7) ... 1478

21.16.31 RSCAN1CFCCk — Transmit/receive FIFO Buffer Configuration and Control Register (k = 0 to 2) ... 1479
21.16.32 RSCAN1CFSTSk — Transmit/receive FIFO Buffer Status Register (k = 0 to 2) ... 1483
21.16.33 RSCAN1CFPCTRk — Transmit/receive FIFO Buffer Pointer Control Register (k = 0 to 2) ... 1486
21.16.34 RSCAN1CFIDk — Transmit/receive FIFO Buffer Access ID Register (k = 0 to 2) ... 1488
21.16.35 RSCAN1CFPTRk — Transmit/receive FIFO Buffer Access Pointer Register (k = 0 to 2) ... 1490
21.16.36 RSCAN1CFDF0k — Transmit/receive FIFO Buffer Access Data Field 0 Register (k = 0 to 2) ... 1492
21.16.37 RSCAN1CFDF1k — Transmit/receive FIFO Buffer Access Data Field 1 Register (k = 0 to 2) ... 1493
21.16.38 RSCAN1FESTS — FIFO Empty Status Register ... 1494
21.16.39 RSCAN1FFSTS — FIFO Full Status Register ... 1495
21.16.40 RSCAN1FMSTS — FIFO Message Lost Status Register ... 1496
21.16.41 RSCAN1RFISTS — Receive FIFO Buffer Interrupt Flag Status Register ... 1497
21.16.42 RSCAN1CFRISTS — Transmit/receive FIFO Buffer Receive Interrupt Flag Status Register ... 1498
21.16.43 RSCAN1CFTISTS — Transmit/receive FIFO Buffer Transmit Interrupt Flag Status Register ... 1499
21.16.44 RSCAN1TMCp — Transmit Buffer Control Register (p = 0 to i × 16 + 15) ... 1500
21.16.45 RSCAN1TMSTSp — Transmit Buffer Status Register (p = 0 to 15) ... 1502
21.16.46 RSCAN1TMTRSTSy — Transmit Buffer Transmit Request Status Register (y = 0) ... 1504
21.16.47 RSCAN1TMTARSTSy — Transmit Buffer Transmit Abort Request Status Register (y = 0) ... 1505
21.16.48 RSCAN1TMTCSTSy — Transmit Buffer Transmit Complete Status Register (y = 0) ... 1506
21.16.49 RSCAN1TMTASTSy — Transmit Buffer Transmit Abort Status Register (y = 0) ... 1507
21.16.50 RSCAN1TMIECy — Transmit Buffer Interrupt Enable Configuration Register (y = 0) ... 1508
21.16.51 RSCAN1TMIDp — Transmit Buffer ID Register (p = 0 to 15) ... 1509
21.16.52 RSCAN1TMPTRp — Transmit Buffer Pointer Register (p = 0 to 15) ... 1511
21.16.53 RSCAN1TMDF0p — Transmit Buffer Data Field 0 Register (p = 0 to 15) ... 1513
21.16.54 RSCAN1TMDF1p — Transmit Buffer Data Field 1 Register (p = 0 to 15) ... 1514
21.16.55 RSCAN1TXQCCm — Transmit Queue Configuration and Control Register (m = 6) ... 1515
21.16.56 RSCAN1TXQSTSm — Transmit Queue Status Register (m = 6) ... 1517
21.16.57 RSCAN1TXQPCTRm — Transmit Queue Pointer Control Register (m = 6) ... 1519
21.16.58 RSCAN1THLCCm — Transmit History Configuration and Control Register (m = 6) ... 1520
21.16.59 RSCAN1THLSTSm — Transmit History Status Register (m = 6) ... 1522
21.16.60 RSCAN1THLACCm — Transmit History Access Register (m = 6) ... 1524
21.16.61 RSCAN1THLPCTRm — Transmit History Pointer Control Register (m = 6) ... 1525
21.16.62 RSCAN1GTSTCFG — Global Test Configuration Register ... 1526
21.16.63 RSCAN1GTSTCTR — Global Test Control Register ... 1527
21.16.64 RSCAN1GLOCKK — Global Lock Key Register ... 1528
21.16.65 RSCAN1RPGACCr — RAM Test Page Access Register (r = 0 to 63) ... 1529
21.17 Interrupt Sources ... 1530
21.18 CAN Modes ... 1533
21.18.1 Global Modes ... 1533
21.18.1.1 Global Stop Mode ... 1535
21.18.1.2 Global Reset Mode ... 1535

21.18.1.3 Global Test Mode ..... 1535
21.18.1.4 Global Operating Mode ..... 1535
21.18.2 Channel Modes ..... 1536
21.18.2.1 Channel Stop Mode ..... 1537
21.18.2.2 Channel Reset Mode ..... 1537
21.18.2.3 Channel Halt Mode ..... 1537
21.18.2.4 Channel Communication Mode ..... 1538
21.18.2.5 Bus Off State ..... 1538
21.19 Reception Function ..... 1541
21.19.1 Data Processing Using the Receive Rule Table ..... 1541
21.19.1.1 Acceptance Filter Processing ..... 1542
21.19.1.2 DLC Filter Processing ..... 1543
21.19.1.3 Routing Processing ..... 1543
21.19.1.4 Label Addition Processing ..... 1543
21.19.1.5 Mirror Function Processing ..... 1543
21.19.1.6 Timestamp ..... 1544
21.20 Transmission Functions ..... 1545
21.20.1 Transmit Priority Determination ..... 1546
21.20.2 Transmission Using Transmit Buffers ..... 1546
21.20.2.1 Transmit Abort Function ..... 1546
21.20.2.2 One-Shot Transmission Function (Retransmission Disabling Function) ..... 1547
21.20.3 Transmission Using FIFO Buffers ..... 1547
21.20.3.1 Interval Transmission Function ..... 1548
21.20.4 Transmission Using Transmit Queues ..... 1550
21.20.5 Transmit History Function ..... 1550
21.21 Test Function ..... 1552
21.21.1 Standard Test Mode ..... 1552
21.21.2 Listen-Only Mode ..... 1552
21.21.3 Self-Test Mode (Loopback Mode) ..... 1553
21.21.3.1 Self-Test Mode 0 (External Loopback Mode) ..... 1553
21.21.3.2 Self-Test Mode 1 (Internal Loopback Mode) ..... 1554
21.21.4 RAM Test ..... 1554
21.22 RS-CAN Setting Procedure ..... 1555
21.22.1 Initial Settings ..... 1555
21.22.1.1 Clock Setting ..... 1556
21.22.1.2 Bit Timing Setting ..... 1556
21.22.1.3 Communication Speed Setting ..... 1557
21.22.1.4 Receive Rule Setting ..... 1558
21.22.1.5 Buffer Setting ..... 1559
21.22.2 Reception Procedure ..... 1561
21.22.2.1 Receive Buffer Reading Procedure ..... 1561
21.22.2.2 FIFO Buffer Reading Procedure ..... 1563
21.22.3 Transmission Procedure ..... 1566
21.22.3.1 Procedure for Transmission from Transmit Buffers ..... 1566
21.22.3.2 Procedure for Transmission from Transmit/Receive FIFO Buffers ..... 1570
21.22.3.3 Procedure for Transmission from the Transmit Queue ..... 1574
21.22.3.4 Transmit History Buffer Reading Procedure ..... 1575

21.22.4 Test Settings ... 1576
21.22.4.1 Self-Test Mode Setting Procedure ... 1576
21.22.4.2 Procedure for Releasing the Protection ... 1577
21.22.4.3 RAM Test Setting Procedure ... 1578
21.23 Detection and Correction of Errors in RS-CAN RAM ... 1579
21.23.1 ECC for the RSCAN1 RAM ... 1579
21.23.2 Interrupt Request ... 1579
21.23.3 List of Registers ... 1580
21.23.4 ECCRCANnCTL — RSCAN1 ECC Control Register ... 1581
21.23.5 ECCRCANnTMC — RSCAN1 ECC Test Mode Control Register ... 1583
21.23.6 ECCRCANnTED — RSCAN1 ECC Encode/Decode Input/Output Replacement Test Register ... 1585
21.23.7 ECCRCANnTRC — RSCAN1 ECC Redundant Bit Data Control Test Register ... 1586
21.23.8 ECCRCANnSYND — RSCAN1 ECC Decode Syndrome Data Register ... 1586
21.23.9 ECCRCANnHORD — RSCAN1 ECC 7-Bit Redundant Bit Data Hold Test Register ... 1587
21.23.10 ECCRCANnECRD — RSCAN1 ECC Encode Test Register ... 1587
21.23.11 ECCRCANnERDB — RSCAN1 ECC Redundant Bit Input/Output Replacement Buffer Register ... 1588
21.23.12 ECCRCAN1AD0 — RSCAN1 ECC Error Address Register 0 ... 1589
21.23.13 SELB_READTEST — ECCREAD Test Select Register ... 1589
21.24 Notes on the RS-CAN Module ... 1590

Section 22 Window Watchdog Timer (WDTA) ... 1591
22.1 Features of RH850/F1K WDTA ... 1591
22.1.1 Number of Units and Channels ... 1591
22.1.2 Register Base Address ... 1591
22.1.3 Clock Supply ... 1591
22.1.4 Interrupt Request ... 1592
22.1.5 Reset Sources ... 1592
22.2 Overview ... 1593
22.2.1 Functional Overview ... 1593
22.2.2 Block Diagram ... 1595
22.3 Registers ... 1596
22.3.1 List of Registers ... 1596
22.3.2 WDTAnWDTE — WDTA Enable Register ... 1597
22.3.3 WDTAnEVAC — WDTA Enable VAC Register ... 1599
22.3.4 WDTAnREF — WDTA Reference Value Register ... 1600
22.3.5 WDTAnMD — WDTA Mode Register ... 1601
22.4 Interrupt Sources ... 1602
22.5 Functions ... 1603
22.5.1 WDTA after Reset Release ... 1603
22.5.1.1 Start Modes ... 1603
22.5.1.2 WDTA Settings after Reset Release ... 1603
22.5.1.3 Default Start Mode Timing ... 1604
22.5.1.4 Software Trigger Start Mode Timing ... 1605
22.5.2 WDTA Trigger ... 1606

22.5.2.1 Calculating an Activation Code when the VAC Function is Used ... 1606
22.5.3 WDTA Error Detection ... 1607
22.5.3.1 WDTA Error Mode ... 1607
22.5.4 75% Interrupt Request Signals ... 1609
22.5.5 Window Function ... 1610

Section 23 OS Timer (OSTM) ... 1611

23.1 Features of RH850/F1K OSTM ... 1611
23.1.1 Number of Units ... 1611
23.1.2 Register Base Address ... 1611
23.1.3 Clock Supply ... 1611
23.1.4 Interrupt Request ... 1612
23.1.5 Reset Sources ... 1612
23.2 Overview ... 1613
23.2.1 Functional Overview ... 1613
23.2.2 Block Diagram ... 1613
23.2.3 Count Clock ... 1614
23.2.4 Interrupt Sources (OSTMTINT) ... 1614
23.3 Registers ... 1615
23.3.1 List of Registers ... 1615
23.3.2 OSTMnCMP — OSTMn Compare Register ... 1616
23.3.3 OSTMnCNT — OSTMn Counter Register ... 1617
23.3.4 OSTMnTE — OSTMn Count Enable Status Register ... 1618
23.3.5 OSTMnTS — OSTMn Count Start Trigger Register ... 1619
23.3.6 OSTMnTT — OSTMn Count Stop Trigger Register ... 1619
23.3.7 OSTMnCTL — OSTMn Control Register ... 1620
23.3.8 OSTMnEMU — OSTMn Emulation Register ... 1621
23.4 Operation ... 1622
23.4.1 Starting and Stopping OSTM ... 1622
23.4.2 Interval Timer Mode ... 1622
23.4.2.1 Basic Operation in Interval Timer Mode ... 1622
23.4.2.2 Operation when OSTMnCMP = 0000 $0000_H$ ... 1625
23.4.2.3 Setting Procedure for Interval Timer Mode ... 1625
23.4.3 Free-Run Compare Mode ... 1626
23.4.3.1 Basic Operation in Free-Run Compare Mode ... 1626
23.4.3.2 Operation when OSTMnCMP = 0000 $0000_H$ ... 1627
23.4.3.3 Setting Procedure for Free-Run Compare Mode ... 1628

Section 24 Timer Array Unit B (TAUB) ... 1629

24.1 Features of RH850/F1K TAUB ... 1629
24.1.1 Number of Units and Channels ... 1629
24.1.2 Register Base Addresses ... 1629
24.1.3 Clock Supply ... 1630
24.1.4 Interrupt Requests ... 1630
24.1.5 Reset Sources ... 1631

24.1.6 External input/output Signals ... 1631
24.2 Overview ... 1634
24.2.1 Functional Overview ... 1634
24.2.2 Terms ... 1635
24.2.3 Functional List of Timer Operations ... 1636
24.2.4 Input/Output Interrupt Request Signals ... 1637
24.2.5 Block Diagram ... 1638
24.2.6 Description of Blocks ... 1639
24.3 Registers ... 1640
24.3.1 List of Registers ... 1640
24.3.2 Details of TAUBn Prescaler Registers ... 1641
24.3.2.1 TAUBnTPS — TAUBn Prescaler Clock Select Register ... 1641
24.3.3 Details of TAUBn Control Registers ... 1644
24.3.3.1 TAUBnCDRm — TAUBn Channel Data Register ... 1644
24.3.3.2 TAUBnCNTm — TAUBn Channel Counter Register ... 1645
24.3.3.3 TAUBnCMORm — TAUBn Channel Mode OS Register ... 1646
24.3.3.4 TAUBnCMURm — TAUBn Channel Mode User Register ... 1649
24.3.3.5 TAUBnCSRm — TAUBn Channel Status Register ... 1650
24.3.3.6 TAUBnCSCm — TAUBn Channel Status Clear Register ... 1650
24.3.3.7 TAUBnTS — TAUBn Channel Start Trigger Register ... 1651
24.3.3.8 TAUBnTE — TAUBn Channel Enable Status Register ... 1651
24.3.3.9 TAUBnTT — TAUBn Channel Stop Trigger Register ... 1652
24.3.4 Details of TAUBn Simultaneous Rewrite Registers ... 1653
24.3.4.1 TAUBnRDE — TAUBn Channel Reload Data Enable Register ... 1653
24.3.4.2 TAUBnRDS — TAUBn Channel Reload Data Control Channel Select Register 1653
24.3.4.3 TAUBnRDM — TAUBn Channel Reload Data Mode Register ... 1654
24.3.4.4 TAUBnRDC — TAUBn Channel Reload Data Control Register ... 1654
24.3.4.5 TAUBnRDT — TAUBn Channel Reload Data Trigger Register ... 1655
24.3.4.6 TAUBnRSF — TAUBn Channel Reload Status Register ... 1655
24.3.5 Details of TAUBn Output Registers ... 1656
24.3.5.1 TAUBnTOE — TAUBn Channel Output Enable Register ... 1656
24.3.5.2 TAUBnTO — TAUBn Channel Output Register ... 1656
24.3.5.3 TAUBnTOM — TAUBn Channel Output Mode Register ... 1657
24.3.5.4 TAUBnTOC — TAUBn Channel Output Configuration Register ... 1657
24.3.5.5 TAUBnTOL — TAUBn Channel Output Active Level Register ... 1658
24.3.6 Details of TAUBn Dead Time Output Registers ... 1659
24.3.6.1 TAUBnTDE — TAUBn Channel Dead Time Output Enable Register ... 1659
24.3.6.2 TAUBnTDL — TAUBn Channel Dead Time Output Level Register ... 1659
24.3.7 TAUBn Emulation Register ... 1660
24.3.7.1 TAUBnEMU — TAUBn Emulation Register ... 1660
24.4 General Operating Procedure ... 1661
24.5 Concepts of Synchronous Channel Operation ... 1662
24.5.1 Rules of Synchronous Channel Operation Function ... 1662
24.5.2 Simultaneous Start and Stop of Synchronous Channel Counters ... 1664
24.5.2.1 Simultaneous Start and Stop within the Same Unit ... 1664
24.6 Simultaneous Rewrite ... 1665
24.6.1 Introduction ... 1665

24.6.2 How to Control Simultaneous Rewrite ... 1666
24.6.2.1 Initial Settings ... 1667
24.6.2.2 Start Counter and Count Operation ... 1667
24.6.2.3 Simultaneous Rewrite ... 1667
24.6.3 Other General Rules of Simultaneous Rewrite ... 1668
24.6.4 Types of Simultaneous Rewrite ... 1669
24.6.4.1 Simultaneous Rewrite when the Master Channel (Re)Starts Counting (Method A) ... 1669
24.6.4.2 Simultaneous Rewrite at the Peak of a Triangular Wave of the Slave Channel (Method B) ... 1670
24.6.4.3 Simultaneous Rewrite when INTTAUBnIm is Generated on an Upper Channel Specified by TAUBnRDC.TAUBnRDCm (Method C1) ... 1672
24.7 Channel Output Modes ... 1674
24.7.1 General Procedures for Specifying a Channel Output Mode ... 1676
24.7.2 Channel Output Modes Controlled Independently by TAUBn Signals ... 1677
24.7.2.1 Independent Channel Output Mode 1 ... 1677
24.7.2.2 Independent Channel Output Mode 2 ... 1677
24.7.3 Channel Output Modes Controlled Synchronously by TAUBn Signals ... 1677
24.7.3.1 Synchronous Channel Output Mode 1 ... 1677
24.7.3.2 Synchronous Channel Output Mode 2 ... 1677
24.7.3.3 Synchronous Channel Output Mode 2 with Dead Time Output ... 1678
24.8 Start Timing in Each Operating Modes ... 1679
24.8.1 Interval Timer Mode, Judge Mode, Capture Mode, Count-Up/-Down Mode, and Count Capture Mode ... 1679
24.8.2 Event Count Mode ... 1680
24.8.3 Other Operating Modes ... 1680
24.9 TAUBTTOUTm Output and INTTAUBnIm Generation when Counter Starts or Restarts ... 1681
24.10 Interrupt Generation upon Overflow ... 1682
24.10.1 Example of Combination of TAUBTTINm Input Pulse Interval Measurement Function and TAUBTTINm Input Interval Timer Function ... 1683
24.10.2 Example of Combination of TAUBTTINm Input Signal Width Measurement Function and Overflow Interrupt Output Function (during TAUBTTINm Width Measurement) ... 1684
24.10.3 Example of Combination of TAUBTTINm Input Position Detection Function and Interval Timer Function ... 1685
24.10.4 Example of Combination of TAUBTTINm Input Period Count Detection Function and Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection) ... 1686
24.11 TAUBTTINm Edge Detection ... 1687
24.12 Independent Channel Operation Functions ... 1688
24.12.1 Interval Timer Function ... 1688
24.12.1.1 Overview ... 1688
24.12.1.2 Equations ... 1688
24.12.1.3 Block Diagram and General Timing Diagram ... 1689
24.12.1.4 Register Settings ... 1690
24.12.1.5 Operating Procedure for Interval Timer Function ... 1692
24.12.1.6 Specific Timing Diagrams ... 1693
24.12.2 TAUBTTINm Input Interval Timer Function ... 1697
24.12.2.1 Overview ... 1697

24.12.2.2 Block Diagram and General Timing Diagram ..... 1698
24.12.2.3 Register Settings ..... 1699
24.12.2.4 Operating Procedure for TAUBTTINm Input Interval Timer Function ..... 1701
24.12.2.5 Specific Timing Diagrams ..... 1702
24.12.3 Clock Divide Function ..... 1703
24.12.3.1 Overview ..... 1703
24.12.3.2 Block Diagram and General Timing Diagram ..... 1704
24.12.3.3 Register Settings ..... 1705
24.12.3.4 Operating Procedure for Clock Divide Function ..... 1706
24.12.3.5 Specific Timing Diagrams ..... 1707
24.12.4 External Event Count Function ..... 1709
24.12.4.1 Overview ..... 1709
24.12.4.2 Equations ..... 1709
24.12.4.3 Block Diagram and General Timing Diagram ..... 1710
24.12.4.4 Register Settings ..... 1711
24.12.4.5 Operating Procedure for External Event Count Function ..... 1712
24.12.4.6 Specific Timing Diagrams ..... 1713
24.12.5 One-Pulse Output Function ..... 1715
24.12.5.1 Overview ..... 1715
24.12.5.2 Equations ..... 1715
24.12.5.3 Block Diagram and General Timing Diagram ..... 1716
24.12.5.4 Register Settings ..... 1717
24.12.5.5 Operating Procedure for One-Pulse Output Function ..... 1718
24.12.6 TAUBTTINm Input Pulse Interval Measurement Function ..... 1719
24.12.6.1 Overview ..... 1719
24.12.6.2 Equations ..... 1720
24.12.6.3 Block Diagram and General Timing Diagram ..... 1721
24.12.6.4 Register Settings ..... 1722
24.12.6.5 Operating Procedure for TAUBTTINm Input Pulse Interval Measurement Function ..... 1723
24.12.6.6 Specific Timing Diagrams: Overflow Behavior ..... 1724
24.12.7 TAUBTTINm Input Signal Width Measurement Function ..... 1728
24.12.7.1 Overview ..... 1728
24.12.7.2 Equations ..... 1729
24.12.7.3 Block Diagram and General Timing Diagram ..... 1729
24.12.7.4 Register Settings ..... 1730
24.12.7.5 Operating Procedure for TAUBTTINm Input Signal Width Measurement Function ..... 1731
24.12.7.6 Specific Timing Diagrams: Overflow Behavior ..... 1732
24.12.8 TAUBTTINm Input Position Detection Function ..... 1736
24.12.8.1 Overview ..... 1736
24.12.8.2 Equations ..... 1736
24.12.8.3 Block Diagram and General Timing Diagram ..... 1737
24.12.8.4 Register Settings ..... 1738
24.12.8.5 Operating Procedure for TAUBTTINm Input Position Detection Function ..... 1739
24.12.8.6 Specific Timing Diagrams ..... 1740
24.12.9 TAUBTTINm Input Period Count Detection Function ..... 1741
24.12.9.1 Overview ..... 1741
24.12.9.2 Equations ..... 1741

24.12.9.3 Block Diagram and General Timing Diagram ... 1742
24.12.9.4 Register Settings ... 1743
24.12.9.5 Operating Procedure for TAUBTTINm Input Period Count Detection Function ... 1744
24.12.9.6 Specific Timing Diagrams ... 1745
24.12.10 TAUBTTINm Input Pulse Interval Judgment Function ... 1746
24.12.10.1 Overview ... 1746
24.12.10.2 Block Diagram and General Timing Diagram ... 1747
24.12.10.3 Register Settings ... 1747
24.12.10.4 Operating Procedure for TAUBTTINm Input Pulse Interval Judgment Function ... 1749
24.12.11 TAUBTTINm Input Signal Width Judgment Function ... 1750
24.12.11.1 Overview ... 1750
24.12.11.2 Block Diagram and General Timing Diagram ... 1751
24.12.11.3 Register Settings ... 1752
24.12.11.4 Operating Procedure for TAUBTTINm Input Signal Width Judgment Function ... 1753
24.12.12 Overflow Interrupt Output Function (during TAUBTTINm Width Measurement) ... 1754
24.12.12.1 Overview ... 1754
24.12.12.2 Block Diagram and General Timing Diagram ... 1755
24.12.12.3 Register Settings ... 1756
24.12.12.4 Operating Procedure for Overflow Interrupt Output Function (during TAUBTTINm Width Measurement) ... 1758
24.12.13 Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection) ... 1759
24.12.13.1 Overview ... 1759
24.12.13.2 Block Diagram and General Timing Diagram ... 1760
24.12.13.3 Register Settings ... 1761
24.12.13.4 Operating Procedure for Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection) ... 1763
24.13 Independent Channel Simultaneous Rewrite Functions ... 1764
24.13.1 Simultaneous Rewrite Trigger Generation Function Type 1 ... 1764
24.13.1.1 Overview ... 1764
24.13.1.2 Equations ... 1765
24.13.1.3 Block Diagram and General Timing Diagram ... 1766
24.13.1.4 Register Settings for The Upper Channel ... 1768
24.13.1.5 Register Settings for the Lower Channel(s) ... 1769
24.13.1.6 Operating Procedure for Simultaneous Rewrite Trigger Generation Function Type 1 ... 1770
24.14 Synchronous Channel Operation Functions ... 1771
24.14.1 PWM Output Function ... 1771
24.14.1.1 Overview ... 1771
24.14.1.2 Equations ... 1772
24.14.1.3 Block Diagram and General Timing Diagram ... 1772
24.14.1.4 Register Settings for the Master Channel ... 1774
24.14.1.5 Register Settings for the Slave Channel(s) ... 1776
24.14.1.6 Operating Procedure for PWM Output Function ... 1778
24.14.1.7 Specific Timing Diagrams ... 1779
24.14.2 One-Shot Pulse Output Function ... 1783
24.14.2.1 Overview ... 1783
24.14.2.2 Equations ... 1784
24.14.2.3 Block Diagram and General Timing Diagram ... 1785

24.14.2.4 Register Settings for the Master Channel ..... 1787
24.14.2.5 Register Settings for the Slave Channel ..... 1789
24.14.2.6 Operating Procedure for One-Shot Pulse Output Function ..... 1791
24.14.2.7 Specific Timing Diagrams ..... 1792
24.14.3 Delay Pulse Output Function ..... 1798
24.14.3.1 Overview ..... 1798
24.14.3.2 Equations ..... 1799
24.14.3.3 Block Diagram and General Timing Diagram ..... 1801
24.14.3.4 Register Settings for the Master Channel ..... 1803
24.14.3.5 Register Settings for Slave Channel 1 ..... 1805
24.14.3.6 Register Settings For Slave Channel 2 ..... 1807
24.14.3.7 Register Settings for Slave Channel 3 ..... 1809
24.14.3.8 Operating Procedure for Delay Pulse Output Function ..... 1810
24.14.3.9 Specific Timing Diagrams ..... 1812
24.14.4 AD Conversion Trigger Output Function Type 1 ..... 1814
24.14.4.1 Overview ..... 1814
24.14.4.2 Block Diagram and General Timing Diagram ..... 1814
24.14.5 Triangle PWM Output Function ..... 1816
24.14.5.1 Overview ..... 1816
24.14.5.2 Equations ..... 1817
24.14.5.3 Block Diagram and General Timing Diagram ..... 1818
24.14.5.4 Register Settings for the Master Channel ..... 1820
24.14.5.5 Register Settings for the Slave Channel(s) ..... 1822
24.14.5.6 Operating Procedure for Triangle PWM Output Function ..... 1824
24.14.5.7 Specific Timing Diagrams ..... 1825
24.14.6 Triangle PWM Output Function with Dead Time ..... 1827
24.14.6.1 Overview ..... 1827
24.14.6.2 Equations ..... 1829
24.14.6.3 Block Diagram and General Timing Diagram ..... 1830
24.14.6.4 Register Settings for the Master Channel ..... 1832
24.14.6.5 Register Settings for Slave Channel 2 ..... 1834
24.14.6.6 Register Settings for Slave Channel 3 ..... 1836
24.14.6.7 Operating Procedure for Triangle PWM Output Function with Dead Time ..... 1838
24.14.6.8 Specific Timing Diagrams ..... 1839
24.14.7 A/D Conversion Trigger Output Function Type 2 ..... 1850
24.14.7.1 Overview ..... 1850
24.14.7.2 Block Diagram and General Timing Diagram ..... 1850

Section 25 Timer Array Unit D (TAUD) ..... 1852

25.1 Features of RH850/F1K TAUD ..... 1852
25.1.1 Number of Units and Channels ..... 1852
25.1.2 Register Base Address ..... 1852
25.1.3 Clock Supply ..... 1853
25.1.4 Interrupt Requests ..... 1853
25.1.5 Reset Sources ..... 1853
25.1.6 External Input/Output Signals ..... 1854
25.1.7 Internal Input/Output Signals ..... 1855
25.1.8 TAUD0 Input Selection ..... 1855

25.1.8.1 List of Registers ... 1857
25.1.8.2 SELB_TAUD0I — TAUDTTINm Input Signal Selection Register ... 1857
25.2 Overview ... 1858
25.2.1 Functional Overview ... 1858
25.2.2 Terms ... 1859
25.2.3 Functional List of Timer Operations ... 1860
25.2.4 TAUD I/O and Interrupt Request Signals ... 1861
25.2.5 Block Diagram ... 1862
25.2.6 Description of Blocks ... 1863
25.3 Registers ... 1864
25.3.1 List of Registers ... 1864
25.3.2 Details of TAUDn Prescaler Registers ... 1865
25.3.2.1 TAUDnTPS — TAUDn Prescaler Clock Select Register ... 1865
25.3.2.2 TAUDnBRS — TAUDn Prescaler Baud Rate Setting Register ... 1868
25.3.3 Details of TAUDn Control Registers ... 1869
25.3.3.1 TAUDnCDRm — TAUDn Channel Data Register ... 1869
25.3.3.2 TAUDnCNTm — TAUDn Channel Counter Register ... 1870
25.3.3.3 TAUDnCMORm — TAUDn Channel Mode OS Register ... 1871
25.3.3.4 TAUDnCMURm — TAUDn Channel Mode User Register ... 1874
25.3.3.5 TAUDnCSRm — TAUDn Channel Status Register ... 1875
25.3.3.6 TAUDnCSCm — TAUDn Channel Status Clear Register ... 1876
25.3.3.7 TAUDnTS — TAUDn Channel Start Trigger Register ... 1876
25.3.3.8 TAUDnTE — TAUDn Channel Enable Status Register ... 1877
25.3.3.9 TAUDnTT — TAUDn Channel Stop Trigger Register ... 1877
25.3.4 Details of TAUDn Simultaneous Rewrite Registers ... 1878
25.3.4.1 TAUDnRDE — TAUDn Channel Reload Data Enable Register ... 1878
25.3.4.2 TAUDnRDS — TAUDn Channel Reload Data Control Channel Select Register 1878
25.3.4.3 TAUDnRDM — TAUDn Channel Reload Data Mode Register ... 1879
25.3.4.4 TAUDnRDC — TAUDn Channel Reload Data Control Register ... 1879
25.3.4.5 TAUDnRDT — TAUDn Channel Reload Data Trigger Register ... 1880
25.3.4.6 TAUDnRSF — TAUDn Channel Reload Status Register ... 1880
25.3.5 Details of TAUDn Output Registers ... 1881
25.3.5.1 TAUDnTOE — TAUDn Channel Output Enable Register ... 1881
25.3.5.2 TAUDnTO — TAUDn Channel Output Register ... 1881
25.3.5.3 TAUDnTOM — TAUDn Channel Output Mode Register ... 1882
25.3.5.4 TAUDnTOC — TAUDn Channel Output Configuration Register ... 1882
25.3.5.5 TAUDnTOL — TAUDn Channel Output Active Level Register ... 1883
25.3.6 Details of TAUDn Dead Time Output Registers ... 1884
25.3.6.1 TAUDnTDE — TAUDn Channel Dead Time Output Enable Register ... 1884
25.3.6.2 TAUDnTDM — TAUDn Channel Dead Time Output Mode Register ... 1884
25.3.6.3 TAUDnTDL — TAUDn Channel Dead Time Output Level Register ... 1885
25.3.7 Details of TAUDn Real-time/Modulation Output Registers ... 1886
25.3.7.1 TAUDnTRE — TAUDn Channel Real-time Output Enable Register ... 1886
25.3.7.2 TAUDnTRC — TAUDn Channel Real-time Output Control Register ... 1886
25.3.7.3 TAUDnTRO — TAUDn Channel Real-time Output Register ... 1887
25.3.7.4 TAUDnTME — TAUDn Channel Modulation Output Enable Register ... 1887
25.3.8 TAUDn Emulation Register ... 1888
25.3.8.1 TAUDnEMU — TAUDn Emulation Register ... 1888

25.4 Operating Procedure ..... 1889
25.5 Concepts of Synchronous Channel Operation ..... 1890
25.5.1 Rules of Synchronous Channel Operation ..... 1890
25.5.2 Simultaneous Start and Stop of Synchronous Channel Counters ..... 1892
25.5.2.1 Simultaneous Start and Stop within the Same Unit ..... 1892
25.5.2.2 Simultaneous Start between the Units ..... 1892
25.6 Simultaneous Rewrite ..... 1893
25.6.1 Overview of Operations ..... 1893
25.6.2 How to Control Simultaneous Rewrite ..... 1895
25.6.2.1 Initial Settings ..... 1896
25.6.2.2 Start Counter and Count Operation ..... 1896
25.6.2.3 Simultaneous Rewrite ..... 1896
25.6.3 Other General Rules of Simultaneous Rewrite ..... 1897
25.6.4 Types of Simultaneous Rewrite ..... 1898
25.6.4.1 Simultaneous Rewrite when the Master Channel (Re)starts Counting (Method A) ..... 1898
25.6.4.2 Simultaneous Rewrite at the Peak of a Triangular Wave of Slave Channel (Method B) ..... 1900
25.6.4.3 Simultaneous Rewrite when INTTAUDnIm is Generated on an Upper Channel Specified by TAUDnRDC.TAUDnRDCm (Method C1) ..... 1901
25.6.4.4 Simultaneous Rewrite when INTTAUDnIm is Generated on an Upper Channel Specified by TAUDnRDC.TAUDnRDCm that in Turn is Triggered by an External Signal (Method C2) ..... 1903
25.7 Channel Output Modes ..... 1905
25.7.1 General Procedures for Specifying a Channel Output Mode ..... 1907
25.7.2 Channel Output Modes Controlled Independently by TAUDn Signals ..... 1908
25.7.2.1 Independent Channel Output Mode 1 ..... 1908
25.7.2.2 Independent Channel Output Mode 1 with Real-Time Output ..... 1908
25.7.2.3 Independent Channel Output Mode 2 ..... 1909
25.7.3 Channel Output Modes Controlled Synchronously by TAUDn Signals ..... 1910
25.7.3.1 Synchronous Channel Output Mode 1 ..... 1910
25.7.3.2 Synchronous Channel Output Mode 1 with Non-Complementary Modulation Output ..... 1910
25.7.3.3 Synchronous Channel Output Mode 2 ..... 1910
25.7.3.4 Synchronous Channel Output Mode 2 with Dead Time Output ..... 1911
25.7.3.5 Synchronous Channel Output Mode 2 with One-Phase PWM Output ..... 1912
25.7.3.6 Synchronous Channel Output Mode 2 with Complementary Modulation Output 1913
25.7.3.7 Synchronous Channel Output Mode 2 with Non-Complementary Modulation Output ..... 1913
25.8 Start Timing in Each Operating Modes ..... 1914
25.8.1 Interval Timer Mode, Judge Mode, Capture Mode, Count-up/-down Mode, and Count Capture Mode ..... 1914
25.8.2 Event Count Mode ..... 1915
25.8.3 Other Operating Modes ..... 1915
25.9 TAUDTTOUTm Output and INTTAUDnIm Generation when Counter Starts or Restarts ..... 1916
25.10 Interrupt Generation upon Overflow ..... 1917
25.10.1 Combination of the TAUDTTINm Input Pulse Interval Measurement Function and the TAUDTTINm Input Interval Timer Function ..... 1918

25.10.2 Combination of the TAUDTTINm Input Signal Width Measurement Function and the Overflow Interrupt Output Function (at Measuring the TAUDTTINm Width) ..... 1919
25.10.3 Combination of the TAUDTTINm Input Position Detection Function and the Interval Timer Function ..... 1920
25.10.4 Combination of the TAUDTTINm Input Period Count Detection Function and the Overflow Interrupt Output Function (at Detecting the TAUDTTINm Input Period Count) ..... 1921
25.11 TAUDTTINm Edge Detection ..... 1922
25.12 Independent Channel Operation Functions ..... 1923
25.12.1 Interval Timer Function ..... 1923
25.12.1.1 Overview ..... 1923
25.12.1.2 Equations ..... 1923
25.12.1.3 Block Diagram and General Timing Diagram ..... 1924
25.12.1.4 Register Settings ..... 1925
25.12.1.5 Operating Procedure for Interval Timer Function ..... 1927
25.12.1.6 Specific Timing Diagrams ..... 1928
25.12.2 TAUDTTINm Input Interval Timer Function ..... 1932
25.12.2.1 Overview ..... 1932
25.12.2.2 Equations ..... 1932
25.12.2.3 Block Diagram and General Timing Diagram ..... 1933
25.12.2.4 Register Settings ..... 1934
25.12.2.5 Operating Procedure for TAUDTTINm Input Interval Timer Function ..... 1936
25.12.2.6 Specific Timing Diagrams ..... 1937
25.12.3 Clock Divide Function ..... 1938
25.12.3.1 Overview ..... 1938
25.12.3.2 Equations ..... 1939
25.12.3.3 Block Diagram and General Timing Diagram ..... 1939
25.12.3.4 Register Settings ..... 1940
25.12.3.5 Operating Procedure for Clock Divide Function ..... 1942
25.12.3.6 Specific Timing Diagrams ..... 1943
25.12.4 External Event Count Function ..... 1945
25.12.4.1 Overview ..... 1945
25.12.4.2 Equations ..... 1945
25.12.4.3 Block Diagram and General Timing Diagram ..... 1946
25.12.4.4 Register Settings ..... 1947
25.12.4.5 Operating Procedure for External Event Count Function ..... 1948
25.12.4.6 Specific Timing Diagrams ..... 1949
25.12.5 Delay Count Function ..... 1951
25.12.5.1 Overview ..... 1951
25.12.5.2 Equations ..... 1951
25.12.5.3 Block Diagram and General Timing Diagram ..... 1952
25.12.5.4 Register Settings ..... 1953
25.12.5.5 Operating Procedure for Delay Count Function ..... 1954
25.12.6 One-Pulse Output Function ..... 1955
25.12.6.1 Overview ..... 1955
25.12.6.2 Equations ..... 1955
25.12.6.3 Block Diagram and General Timing Diagram ..... 1956
25.12.6.4 Register Settings ..... 1957
25.12.6.5 Operating Procedure for One-Pulse Output Function ..... 1959

25.12.7 TAUDTTINm Input Pulse Interval Measurement Function ..... 1960
25.12.7.1 Overview ..... 1960
25.12.7.2 Equations ..... 1961
25.12.7.3 Block Diagram and General Timing Diagram ..... 1962
25.12.7.4 Register Settings ..... 1963
25.12.7.5 Operating Procedure for TAUDTTINm Input Pulse Interval Measurement Function ..... 1964
25.12.7.6 Specific Timing Diagrams: Overflow Operation ..... 1965
25.12.8 TAUDTTINm Input Signal Width Measurement Function ..... 1968
25.12.8.1 Overview ..... 1968
25.12.8.2 Equations ..... 1969
25.12.8.3 Block Diagram and General Timing Diagram ..... 1969
25.12.8.4 Register Settings ..... 1971
25.12.8.5 Operating Procedure for TAUDTTINm Input Signal Width Measurement Function ..... 1972
25.12.8.6 Specific Timing Diagrams: Overflow Operation ..... 1973
25.12.9 TAUDTTINm Input Position Detection Function ..... 1977
25.12.9.1 Overview ..... 1977
25.12.9.2 Equations ..... 1977
25.12.9.3 Block Diagram and General Timing Diagram ..... 1978
25.12.9.4 Register Settings ..... 1979
25.12.9.5 Operating Procedure for TAUDTTINm Input Position Detection Function ..... 1980
25.12.9.6 Specific Timing Diagrams ..... 1981
25.12.10 TAUDTTINm Input Period Count Detection Function ..... 1982
25.12.10.1 Overview ..... 1982
25.12.10.2 Equations ..... 1982
25.12.10.3 Block Diagram and General Timing Diagram ..... 1983
25.12.10.4 Register Settings ..... 1984
25.12.10.5 Operating Procedure for TAUDTTINm Input Period Count Detection Function ..... 1985
25.12.10.6 Specific Timing Diagrams ..... 1986
25.12.11 TAUDTTINm Input Pulse Interval Judgment Function ..... 1987
25.12.11.1 Overview ..... 1987
25.12.11.2 Block Diagram and General Timing Diagram ..... 1988
25.12.11.3 Register Settings ..... 1989
25.12.11.4 Operating Procedure for TAUDTTINm Input Pulse Interval Judgment Function ..... 1990
25.12.12 TAUDTTINm Input Signal Width Judgment Function ..... 1991
25.12.12.1 Overview ..... 1991
25.12.12.2 Block Diagram and General Timing Diagram ..... 1992
25.12.12.3 Register Settings ..... 1993
25.12.12.4 Operating Procedure for TAUDTTINm Input Signal Width Judgment Function ..... 1994
25.12.13 Overflow Interrupt Output Function (during TAUDTTINm Width Measurement) ..... 1995
25.12.13.1 Overview ..... 1995
25.12.13.2 Block Diagram and General Timing Diagram ..... 1996
25.12.13.3 Register Settings ..... 1997
25.12.13.4 Operating Procedure for Overflow Interrupt Output Function (during TAUDTTINm Width Measurement) ..... 1998
25.12.14 Overflow Interrupt Output Function (during TAUDTTINm Input Period Count Detection) ..... 1999
25.12.14.1 Overview ..... 1999
25.12.14.2 Block Diagram and General Timing Diagram ..... 2000

25.12.14.3 Register Settings ..... 2001
25.12.14.4 Operating Procedure for Overflow Interrupt Output Function (during TAUDTTINm Input Period Count Detection) ..... 2002
25.12.15 One-Phase PWM Output Function ..... 2003
25.12.15.1 Overview ..... 2003
25.12.15.2 Block Diagram and General Timing Diagram ..... 2004
25.12.15.3 Register Settings for Lower Channels ..... 2006
25.12.15.4 Register Settings for Upper Channels ..... 2008
25.12.15.5 Operating Procedure for One-phase PWM Output Function ..... 2009
25.13 Independent Channel Real-Time Functions ..... 2010
25.13.1 Real-Time Output Function Type 1 ..... 2010
25.13.1.1 Overview ..... 2010
25.13.1.2 Equations ..... 2011
25.13.1.3 Block Diagram and General Timing Diagram ..... 2011
25.13.1.4 Register Settings for Upper Channels ..... 2013
25.13.1.5 Register Settings for Lower Channels ..... 2015
25.13.1.6 Operating Procedure for Real-Time Output Function Type 1 ..... 2016
25.13.1.7 Specific Timing Diagrams ..... 2017
25.13.2 Real-Time Output Function Type 2 ..... 2018
25.13.2.1 Overview ..... 2018
25.13.2.2 Block Diagram and General Timing Diagram ..... 2019
25.13.2.3 Register Settings for Upper Channels ..... 2021
25.13.2.4 Register Settings for Lower Channels ..... 2023
25.13.2.5 Operating Procedure for Real-Time Output Function Type 2 ..... 2024
25.13.2.6 Specific Timing Diagrams ..... 2025
25.14 Independent Channel Simultaneous Rewrite Functions ..... 2026
25.14.1 Simultaneous Rewrite Trigger Generation Function Type 1 ..... 2026
25.14.1.1 Overview ..... 2026
25.14.1.2 Equations ..... 2027
25.14.1.3 Block Diagram and General Timing Diagram ..... 2028
25.14.1.4 Register Settings for Upper Channels ..... 2030
25.14.1.5 Register Settings for Lower Channels ..... 2031
25.14.1.6 Operating Procedure for Simultaneous Rewrite Trigger Generation Function Type 1 ..... 2032
25.14.2 Simultaneous Rewrite Trigger Generation Function Type 2 ..... 2033
25.14.2.1 Overview ..... 2033
25.14.2.2 Block Diagram and General Timing Diagram ..... 2034
25.14.2.3 Register Settings for Upper Channels ..... 2036
25.14.2.4 Register Settings for Lower Channels ..... 2038
25.14.2.5 Operating Procedure for Simultaneous Rewrite Trigger Generation Function Type 2 ..... 2039
25.15 Synchronous Channel Operation Functions ..... 2041
25.15.1 PWM Output Function ..... 2041
25.15.1.1 Overview ..... 2041
25.15.1.2 Equations ..... 2042
25.15.1.3 Block Diagram and General Timing Diagram ..... 2042
25.15.1.4 Register Settings for the Master Channel ..... 2044
25.15.1.5 Register Settings for Slave Channels ..... 2046
25.15.1.6 Operating Procedure for PWM Output Function ..... 2048

25.15.1.7 Specific Timing Diagrams ..... 2049
25.15.2 One-Shot Pulse Output Function ..... 2052
25.15.2.1 Overview ..... 2052
25.15.2.2 Equations ..... 2053
25.15.2.3 Block Diagram and General Timing Diagram ..... 2053
25.15.2.4 Register Settings for the Master Channel ..... 2055
25.15.2.5 Register Settings for Slave Channels ..... 2057
25.15.2.6 Operating Procedure for One-Shot Pulse Output Function ..... 2059
25.15.2.7 Specific Timing Diagrams ..... 2060
25.15.3 Trigger Start PWM Output Function ..... 2064
25.15.3.1 Overview ..... 2064
25.15.3.2 Equations ..... 2065
25.15.3.3 Block Diagram and General Timing Diagram ..... 2065
25.15.3.4 Register Settings for the Master Channel ..... 2067
25.15.3.5 Register Settings for Slave Channels ..... 2069
25.15.3.6 Operating Procedure for Trigger Start PWM Output Function ..... 2071
25.15.3.7 Specific Timing Diagrams ..... 2072
25.15.4 Delay Pulse Output Function ..... 2075
25.15.4.1 Overview ..... 2075
25.15.4.2 Equations ..... 2076
25.15.4.3 Block Diagram and General Timing Diagram ..... 2077
25.15.4.4 Register Settings for the Master Channel ..... 2079
25.15.4.5 Register Settings for Slave Channel 1 ..... 2081
25.15.4.6 Register Settings for Slave Channel 2 ..... 2083
25.15.4.7 Register Settings for Slave Channel 3 ..... 2085
25.15.4.8 Operating Procedure for Delay Pulse Output Function ..... 2087
25.15.4.9 Specific Timing Diagrams ..... 2089
25.15.5 Offset Trigger Output Function ..... 2091
25.15.5.1 Overview ..... 2091
25.15.5.2 Equations ..... 2092
25.15.5.3 Block Diagram and General Timing Diagram ..... 2092
25.15.5.4 Register Settings for the Master Channel ..... 2094
25.15.5.5 Register Settings for Slave Channels ..... 2096
25.15.5.6 Operating Procedure for Offset Trigger Output Function ..... 2098
25.15.5.7 Specific Timing Diagrams ..... 2099
25.15.6 A/D Conversion Trigger Output Function Type 1 ..... 2101
25.15.6.1 Overview ..... 2101
25.15.6.2 Block Diagram and General Timing Diagram ..... 2101
25.15.7 Triangle PWM Output Function ..... 2103
25.15.7.1 Overview ..... 2103
25.15.7.2 Equations ..... 2104
25.15.7.3 Block Diagram and General Timing Diagram ..... 2105
25.15.7.4 Register Settings for the Master Channel ..... 2107
25.15.7.5 Register Settings for Slave Channels ..... 2109
25.15.7.6 Operating Procedure for Triangle PWM Output Function ..... 2111
25.15.7.7 Specific Timing Diagrams ..... 2112
25.15.8 Triangle PWM Output Function with Dead Time ..... 2114
25.15.8.1 Overview ..... 2114
25.15.8.2 Equations ..... 2116

25.15.8.3 Block Diagram and General Timing Diagram ..... 2117
25.15.8.4 Register Settings for the Master Channel ..... 2119
25.15.8.5 Register Settings for Slave Channel 2 ..... 2121
25.15.8.6 Register Settings for Slave Channel 3 ..... 2123
25.15.8.7 Operating Procedure for Triangle PWM Output Function with Dead Time ..... 2125
25.15.8.8 Specific Timing Diagrams ..... 2126
25.15.9 A/D Conversion Trigger Output Function Type 2 ..... 2128
25.15.9.1 Overview ..... 2128
25.15.9.2 Block Diagram and General Timing Diagram ..... 2128
25.15.10 Interrupt Request Signals Culling Function ..... 2130
25.15.10.1 Overview ..... 2130
25.15.10.2 Equations ..... 2131
25.15.10.3 Block Diagram and General Timing Diagram ..... 2131
25.15.10.4 Register Settings for the Master Channel ..... 2133
25.15.10.5 Register Settings for the Slave Channel ..... 2135
25.15.10.6 Operating Procedure for Interrupt Request Signals Culling Function ..... 2136
25.15.10.7 Specific Timing Diagrams ..... 2137
25.16 Synchronous Non-Complementary and Complementary Modulation Output Functions ..... 2138
25.16.1 Non-Complementary Modulation Output Function Type 1 ..... 2138
25.16.1.1 Overview ..... 2138
25.16.1.2 Equations ..... 2140
25.16.1.3 Block Diagram and General Timing Diagram ..... 2141
25.16.1.4 Register Settings for the Master Channel ..... 2143
25.16.1.5 Register Settings for Slave Channel 1 ..... 2145
25.16.1.6 Register Settings for Slave Channels 2 to 7 ..... 2147
25.16.1.7 Operating Procedure for Non-Complementary Modulation Output Function Type 1 ..... 2149
25.16.1.8 Specific Timing Diagrams ..... 2151
25.16.2 Non-Complementary Modulation Output Function Type 2 ..... 2152
25.16.2.1 Overview ..... 2152
25.16.2.2 Equations ..... 2154
25.16.2.3 Block Diagram and General Timing Diagram ..... 2155
25.16.2.4 Register Settings for the Master Channel ..... 2157
25.16.2.5 Register Settings for Slave Channel 1 ..... 2159
25.16.2.6 Register settings for slave channels 2 to 7 ..... 2161
25.16.2.7 Operating Procedure for Non-Complementary Modulation Output Function Type 2 ..... 2163
25.16.2.8 Specific Timing Diagrams ..... 2165
25.16.3 Complementary Modulation Output Function ..... 2166
25.16.3.1 Overview ..... 2166
25.16.3.2 Equations ..... 2169
25.16.3.3 Block Diagram and General Timing Diagram ..... 2170
25.16.3.4 Register Settings for the Master Channel ..... 2172
25.16.3.5 Register Settings for Slave Channel 1 ..... 2174
25.16.3.6 Register settings for slave channels 2, 4, and 6 ..... 2176
25.16.3.7 Register settings for slave channels 3, 5, and 7 ..... 2178
25.16.3.8 Operating Procedure for Complementary Modulation Output Function ..... 2180
25.16.3.9 Specific Timing Diagrams ..... 2182

Section 26 Timer Array Unit J (TAUJ) ..... 2184
26.1 Features of RH850/F1K TAUJ ..... 2184
26.1.1 Number of Units ..... 2184
26.1.2 Register Base Address ..... 2184
26.1.3 Clock Supply ..... 2185
26.1.4 Interrupt Requests ..... 2185
26.1.5 Reset Sources ..... 2185
26.1.6 External Input/Output Signals ..... 2186
26.1.7 Internal Input/Output Signals ..... 2186
26.1.8 TAUJ0 Input Selection ..... 2186
26.1.8.1 List of Registers ..... 2188
26.1.8.2 SELB_TAUJ0I — TAUJTTINm Input Signal Selection Register ..... 2188
26.2 Overview ..... 2189
26.2.1 Functional Overview ..... 2189
26.2.2 Terms ..... 2190
26.2.3 Functional List of Timer Operations ..... 2191
26.2.4 TAUJ I/O and Interrupt Request Signals ..... 2191
26.2.5 Block Diagram ..... 2192
26.2.6 Description of Block Diagram ..... 2193
26.3 Registers ..... 2194
26.3.1 List of Registers ..... 2194
26.3.2 Details of TAUJn Prescaler Registers ..... 2195
26.3.2.1 TAUJnTPS — TAUJn Prescaler Clock Select Register ..... 2195
26.3.2.2 TAUJnBRS — TAUJn Prescaler Baud Rate Setting Register ..... 2198
26.3.3 Details of TAUJn Control Registers ..... 2199
26.3.3.1 TAUJnCDRm — TAUJn Channel Data Register ..... 2199
26.3.3.2 TAUJnCNTm — TAUJn Channel Counter Register ..... 2200
26.3.3.3 TAUJnCMORm — TAUJn Channel Mode OS Register ..... 2202
26.3.3.4 TAUJnCMURm — TAUJn Channel Mode User Register ..... 2205
26.3.3.5 TAUJnCSRm — TAUJn Channel Status Register ..... 2206
26.3.3.6 TAUJnCSCm — TAUJn Channel Status Clear Trigger Register ..... 2206
26.3.3.7 TAUJnTS — TAUJn Channel Start Trigger Register ..... 2207
26.3.3.8 TAUJnTE — TAUJn Channel Enable Status Register ..... 2207
26.3.3.9 TAUJnTT — TAUJn Channel Stop Trigger Register ..... 2208
26.3.4 Details of TAUJn Simultaneous Rewrite Register ..... 2209
26.3.4.1 TAUJnRDE — TAUJn Channel Reload Data Enable Register ..... 2209
26.3.4.2 TAUJnRDM — TAUJn Channel Reload Data Mode Register ..... 2209
26.3.4.3 TAUJnRDT — TAUJn Channel Reload Data Trigger Register ..... 2210
26.3.4.4 TAUJnRSF — TAUJn Channel Reload Status Register ..... 2210
26.3.5 Details of TAUJn Output Registers ..... 2211
26.3.5.1 TAUJnTOE — TAUJn Channel Output Enable Register ..... 2211
26.3.5.2 TAUJnTO — TAUJn Channel Output Register ..... 2211
26.3.5.3 TAUJnTOM — TAUJn Channel Output Mode Register ..... 2212
26.3.5.4 TAUJnTOC — TAUJn Channel Output Configuration Register ..... 2213
26.3.5.5 TAUJnTOL — TAUJn Channel Output Active Level Register ..... 2214
26.3.6 TAUJn Emulation Register ..... 2215
26.3.6.1 TAUJnEMU — TAUJn Emulation Register ..... 2215

26.4 Operating Procedure ..... 2216
26.5 Concepts of Synchronous Channel Operation Function ..... 2217
26.5.1 Rules of Synchronous Channel Operation Function ..... 2217
26.5.2 Simultaneous Start and Stop of Synchronous Channel Counters ..... 2219
26.5.2.1 Simultaneous Start and Stop within a TAUJ Unit ..... 2219
26.5.2.2 Simultaneous Start between TAUJ Units ..... 2219
26.6 Simultaneous Rewrite ..... 2220
26.6.1 How to Control Simultaneous Rewrite ..... 2220
26.6.1.1 Initial Settings ..... 2221
26.6.1.2 Start Counter and Count Operation ..... 2221
26.6.1.3 Simultaneous Rewrite ..... 2221
26.6.2 Other General Rules for Simultaneous Rewrite ..... 2221
26.6.3 Simultaneous Rewrite Procedure ..... 2222
26.7 Channel Output Modes ..... 2224
26.7.1 General Procedures for Specifying a Channel Output Mode ..... 2226
26.7.2 Channel Output Modes Controlled Independently by TAUJn Signals ..... 2227
26.7.2.1 Independent Channel Output Mode 1 ..... 2227
26.7.3 Channel Output Modes Controlled Synchronously by TAUJn Signals ..... 2227
26.7.3.1 Synchronous Channel Output Mode 1 ..... 2227
26.8 Start Timing in Each Operating Modes ..... 2228
26.8.1 Interval Timer Mode, Capture Mode, and Count Capture Mode ..... 2228
26.8.2 Other Operating Modes ..... 2229
26.9 TAUJTTOUTm Output and INTTAUJnIm Generation when Counter Starts or Restarts ..... 2230
26.10 Interrupt Generation upon Overflow ..... 2231
26.10.1 Combination of the TAUJTTINm Input Position Detection Function and the Interval Timer Function ..... 2232
26.11 TAUJTTINm Edge Detection ..... 2233
26.12 Independent Channel Operation Functions ..... 2234
26.12.1 Interval Timer Function ..... 2234
26.12.1.1 Overview ..... 2234
26.12.1.2 Equations ..... 2234
26.12.1.3 Block Diagram and General Timing Diagram ..... 2235
26.12.1.4 Register Settings ..... 2236
26.12.1.5 Operating Procedure for Interval Timer Function ..... 2238
26.12.1.6 Specific Timing Diagrams ..... 2239
26.12.2 TAUJTTINm Input Interval Timer Function ..... 2241
26.12.2.1 Overview ..... 2241
26.12.2.2 Equations ..... 2241
26.12.2.3 Block Diagram and General Timing Diagram ..... 2242
26.12.2.4 Register Settings ..... 2243
26.12.2.5 Operating Procedure for TAUJTTINm Input Interval Timer Function ..... 2245
26.12.2.6 Specific Timing Diagrams ..... 2246
26.12.3 TAUJTTINm Input Pulse Interval Measurement Function ..... 2247
26.12.3.1 Overview ..... 2247
26.12.3.2 Equations ..... 2248
26.12.3.3 Block Diagram and General Timing Diagram ..... 2248

26.12.3.4 Register Settings ..... 2250
26.12.3.5 Operating Procedure for TAUJTTINm Input Pulse Interval Measurement Function ..... 2251
26.12.3.6 Specific Timing Diagrams: Overflow Behavior ..... 2252
26.12.4 TAUJTTINm Input Signal Width Measurement Function ..... 2256
26.12.4.1 Overview ..... 2256
26.12.4.2 Equations ..... 2257
26.12.4.3 Block Diagram and General Timing Diagram ..... 2257
26.12.4.4 Register Settings ..... 2258
26.12.4.5 Operating Procedure for TAUJTTINm Input Signal Width Measurement Function ..... 2259
26.12.4.6 Specific Timing Diagrams: Overflow Behavior ..... 2260
26.12.5 TAUJTTINm Input Position Detection Function ..... 2264
26.12.5.1 Overview ..... 2264
26.12.5.2 Equations ..... 2264
26.12.5.3 Block Diagram and General Timing Diagram ..... 2265
26.12.5.4 Register Settings ..... 2266
26.12.5.5 Operating Procedure for TAUJTTINm Input Position Detection Function ..... 2267
26.12.5.6 Specific Timing Diagrams ..... 2268
26.12.6 TAUJTTINm Input Period Count Detection Function ..... 2269
26.12.6.1 Overview ..... 2269
26.12.6.2 Equations ..... 2269
26.12.6.3 Block Diagram and General Timing Diagram ..... 2270
26.12.6.4 Register Settings ..... 2271
26.12.6.5 Operating Procedure for TAUJTTINm Input Period Count Detection Function ..... 2272
26.12.6.6 Specific Timing Diagrams ..... 2273
26.12.7 Overflow Interrupt Output Function (during TAUJTTINm Width Measurement) ..... 2274
26.12.7.1 Overview ..... 2274
26.12.7.2 Block Diagram and General Timing Diagram ..... 2275
26.12.7.3 Register Settings ..... 2276
26.12.7.4 Operating Procedure for Overflow Interrupt Output Function (during TAUJTTINm Width Measurement) ..... 2277
26.12.8 Overflow Interrupt Output Function (during TAUJTTINm Input Period Count Detection) ..... 2278
26.12.8.1 Overview ..... 2278
26.12.8.2 Block Diagram and General Timing Diagram ..... 2279
26.12.8.3 Register Settings ..... 2280
26.12.8.4 Operating Procedure for Overflow Interrupt Output Function (during TAUJTTINm Input Period Count Detection) ..... 2281
26.13 Synchronous Channel Operation Functions ..... 2282
26.13.1 PWM Output Function ..... 2282
26.13.1.1 Overview ..... 2282
26.13.1.2 Equations ..... 2284
26.13.1.3 Block Diagram and General Timing Diagram ..... 2284
26.13.1.4 Register Settings for the Master Channel ..... 2286
26.13.1.5 Register Settings for the Slave Channel(s) ..... 2288
26.13.1.6 Operating Procedure for PWM Output Function ..... 2290
26.13.1.7 Specific Timing Diagrams ..... 2291

Section 27 Real-Time Clock (RTCA) ..... 2294
27.1 Features of RH850/F1K RTCA ..... 2294
27.1.1 Number of Units and Channels ..... 2294
27.1.2 Register Base Address ..... 2294
27.1.3 Clock Supply ..... 2294
27.1.4 Interrupt Requests ..... 2295
27.1.5 Reset Sources ..... 2295
27.1.6 External Input/Output Signals ..... 2295
27.2 Overview ..... 2296
27.2.1 Functional Overview ..... 2296
27.2.2 Block Diagram ..... 2296
27.2.3 Description of Blocks ..... 2297
27.3 Registers ..... 2298
27.3.1 List of Registers ..... 2298
27.3.2 Details of RTCA Control Registers ..... 2299
27.3.2.1 RTCAnCTL0 — RTCA Control Register 0 ..... 2299
27.3.2.2 RTCAnCTL1 — RTCA Control Register 1 ..... 2300
27.3.2.3 RTCAnCTL2 — RTCA Control Register 2 ..... 2301
27.3.3 Details of RTCA Sub-Counter Registers ..... 2303
27.3.3.1 RTCAnSUBC — RTCA Sub-Count Register ..... 2303
27.3.3.2 RTCAnSRBU — RTCA Sub-Count Register Read Buffer ..... 2304
27.3.3.3 RTCAnSUBU — RTCA Clock Error Correction Register ..... 2305
27.3.3.4 RTCAnSCMP — RTCA Sub-Counter Compare Register ..... 2306
27.3.4 Details of RTCA Clock Counter and Buffer Registers ..... 2307
27.3.4.1 RTCAnSECC — RTCA Seconds Count Register ..... 2307
27.3.4.2 RTCAnSEC — RTCA Seconds Count Buffer Register ..... 2308
27.3.4.3 RTCAnMINC — RTCA Minutes Count Register ..... 2309
27.3.4.4 RTCAnMIN — RTCA Minutes Count Buffer Register ..... 2310
27.3.4.5 RTCAnHOURC — RTCA Hours Count Register ..... 2311
27.3.4.6 RTCAnHOUR — RTCA Hours Count Buffer Register ..... 2313
27.3.4.7 RTCAnWEEKC — RTCA Day of the Week Count Register ..... 2314
27.3.4.8 RTCAnWEEK — RTCA Day of the Week Count Buffer Register ..... 2315
27.3.4.9 RTCAnDAYC — RTCA Day of the Month Count Register ..... 2316
27.3.4.10 RTCAnDAY — RTCA Day of the Month Count Buffer Register ..... 2317
27.3.4.11 RTCAnMONC — RTCA Month Count Register ..... 2318
27.3.4.12 RTCAnMONTH — RTCA Month Count Buffer Register ..... 2319
27.3.4.13 RTCAnYEARC — RTCA Year Count Register ..... 2320
27.3.4.14 RTCAnYEAR — RTCA Year Count Buffer Register ..... 2321
27.3.5 Details of RTCA Special Counter and Buffer Registers ..... 2322
27.3.5.1 RTCAnTIMEC — RTCA Time Count Register ..... 2322
27.3.5.2 RTCAnTIME — RTCA Time Count Buffer Register ..... 2323
27.3.5.3 RTCAnCALC — RTCA Calendar Count Register ..... 2324
27.3.5.4 RTCAnCAL — RTCA Calendar Count Buffer Register ..... 2325
27.3.6 Details of RTCA Alarm Setting Registers ..... 2326
27.3.6.1 RTCAnALM — RTCA Alarm Minute Setting Register ..... 2326
27.3.6.2 RTCAnALH — RTCA Alarm Hour Setting Register ..... 2327
27.3.6.3 RTCAnALW — RTCA Alarm Day of the Week Setting Register ..... 2328

27.3.7 RTCA Emulation Register ... 2329
27.3.7.1 RTCAnEMU — RTCA Emulation Register ... 2329
27.4 Operation ... 2330
27.4.1 Clock Counter Format ... 2331
27.4.2 Fixed Interval Interrupt Function ... 2331
27.4.3 Alarm Interrupt Function ... 2331
27.4.4 Clock Error Correction ... 2332
27.4.4.1 Setting the Correction Value and the Operator ... 2334
27.4.4.2 Impact of the Repetition Interval ... 2334
27.4.4.3 Sample Settings ... 2335
27.5 Procedures for Setup, Writing and Reading ... 2337
27.5.1 Initial Setting of the RTCA ... 2337
27.5.1.1 RTCA Stop Procedure ... 2337
27.5.1.2 RTCA Initialization Procedure ... 2338
27.5.2 Updating Clock Counters ... 2339
27.5.3 Reading Clock Counters ... 2340
27.5.3.1 Procedure for Reading Count Buffer Registers ... 2340
27.5.3.2 Procedure for Reading Counter Registers Directly ... 2342
27.5.4 Reading RTCAnSRBU ... 2343
27.5.5 Writing to RTCAnSUBU ... 2344
27.5.6 Writing to RTCAnSCMP ... 2345
27.6 Timing Diagrams ... 2346
27.6.1 Timing of Counter Start ... 2346
27.6.2 Timing of Clock Counter Update while Counter Is Enabled ... 2347
27.6.3 Timing of Sub-Counter Read Buffer Reading while Counter is Enabled ... 2348

Section 28 Encoder Timer (ENCA) ... 2349

28.1 Features of RH850/F1K ENCA ... 2349
28.1.1 Number of Units and Channels ... 2349
28.1.2 Register Base Address ... 2349
28.1.3 Clock Supply ... 2350
28.1.4 Interrupt Requests ... 2350
28.1.5 Reset Sources ... 2350
28.1.6 External Input/Output Signals ... 2350
28.1.7 Internal Input/Output Signals ... 2351
28.2 Overview ... 2352
28.2.1 Functional Overview ... 2352
28.2.2 Block Diagram ... 2353
28.3 Registers ... 2354
28.3.1 List of Registers ... 2354
28.3.2 ENCAnCTL — ENCAn Control Register ... 2355
28.3.3 ENCAnIOC0 — ENCAn I/O Control Register 0 ... 2357
28.3.4 ENCAnIOC1 — ENCAn I/O Control Register 1 ... 2358
28.3.5 ENCAnFLG — ENCAn Status Flag Register ... 2360
28.3.6 ENCAnFGC — ENCAn Status Flag Clear Register ... 2361

28.3.7 ENCAnCCR0 — ENCAn Capture/Compare Register 0 ..... 2362
28.3.8 ENCAnCCR1 — ENCAn Capture/Compare Register 1 ..... 2363
28.3.9 ENCAnCNT — ENCAn Counter Register ..... 2364
28.3.10 ENCAnTE — ENCAn Timer Enable Status Register ..... 2365
28.3.11 ENCAnTS — ENCAn Timer Start Trigger Register ..... 2366
28.3.12 ENCAnTT — ENCAn Timer Stop Trigger Register ..... 2367
28.3.13 ENCAnEMU — ENCAn Emulation Register ..... 2368
28.4 Operation ..... 2369
28.4.1 Timer Counter Operation ..... 2369
28.4.2 Up/Down Control of Timer Counter ..... 2371
28.4.2.1 When the ENCAnUDS1, ENCAnUDS0 Bits in the ENCAnCTL Register = $00_B$ ..... 2371
28.4.2.2 When the ENCAnUDS1, ENCAnUDS0 Bits in the ENCAnCTL Register = $01_B$ ..... 2372
28.4.2.3 When the ENCAnUDS1, ENCAnUDS0 Bits in the ENCAnCTL Register = $10_B$ ..... 2373
28.4.2.4 When ENCAnUDS1, ENCAnUDS0 Bits in the ENCAnCTL Register = $11_B$ ..... 2374
28.4.3 Timer Counter Clear Control by Encoder Input ..... 2375
28.4.3.1 Clearing Method when ENCAnSCE = 0 ..... 2375
28.4.3.2 Clearing Method when ENCAnSCE = 1 ..... 2375
28.4.4 Functions of ENCAnCCR0 ..... 2376
28.4.4.1 Compare Function ..... 2376
28.4.4.2 Capture Function ..... 2376
28.4.5 Functions of ENCAnCCR1 ..... 2377
28.4.5.1 Compare Function ..... 2377
28.4.5.2 Capture Function ..... 2378
28.4.5.3 Timer Counter Clearing upon Compare Register Match ..... 2379
28.4.6 Startup/Stop of Timer Counter ..... 2379
28.4.6.1 Startup of Timer ..... 2379
28.4.6.2 Stop of Timer ..... 2379
28.5 ENCAn Setting Sequences ..... 2380
28.5.1 ENCAn Setting Procedure ..... 2380
28.5.1.1 Initial Setting Procedure for the Counter ..... 2380
28.5.1.2 Initial Setting Procedure for Counter Clear ..... 2381
28.5.1.3 Setting Procedure for ENCAnCCR0 Register ..... 2381
28.5.1.4 Setting Procedure for ENCAnCCR1 Register ..... 2381
28.6 Timing Chart ..... 2382
28.6.1 Overflow Occurrence and Overflow Flag Clear Operation ..... 2382
28.6.2 Underflow Occurrence and Underflow Flag Clear Operation ..... 2383
28.6.3 Count Clearing and Capture Operation by Encoder Clear Input (ENCAnEC Pin) ..... 2384
28.6.4 Conflict between Overflow Occurrence and Clear Operation by Encoder Clear Input (ENCAnEC Pin) ..... 2385
28.6.5 Conflict between Underflow Occurrence and Clear Operation by Encoder Clear Input (ENCAnEC Pin) ..... 2386
28.6.6 Overflow Operation Immediately after Startup ..... 2387
28.6.7 Underflow Operation Immediately after Startup ..... 2388
28.6.8 Using the ENCAnLDE Function Immediately after Startup ..... 2389
28.6.9 ENCAnLDE Function (Loading Count Value) ..... 2390
28.6.10 Conflict between ENCAnLDE Function (Loading Counter Value) and Rewrite of ENCAnCCR0 Register ..... 2392

28.6.11 Conflict between ENCAnLDE Function (Loading Counter Value) and Clear Operation by Encoder Clear Input (ENCAnEC Pin) ..... 2393
28.6.12 Up-count after Conflict between ENCAnLDE Function (Loading Counter Value) and Clear Operation by Encoder Clear Input ..... 2395
28.6.13 Capture Operation between Count Clocks (ENCAnCCR1) ..... 2396
28.6.14 Capture Operation between Count Clocks (ENCAnCCR0) ..... 2397
28.6.15 Encoder Operation when Compare Match Clear Control is Enabled and ENCAnCTS = 0 ..... 2398
28.6.16 Encoder Operation when Compare Match Clear Control is Enabled and ENCAnCTS = 1 ..... 2399
28.6.17 Encoder Operation when Compare Match Clear Control is Disabled ..... 2400
28.6.18 Capture Operation Performed upon Clearing by ENCAnEC, ENCAnE0, ENCAnE1 when ENCAnSCE = 1 ..... 2401
28.6.18.1 Accompanying capture operation ..... 2401
28.6.18.2 When the Timing of the ENCAnEC Input is Later than that of the ENCAnE1 Input during Up-count (When ENCAnACL = 1, ENCAnBCL = 0, ENCAnZCL = 1, and ENCAnUDS[1:0] = $11_B$) ..... 2402
28.6.18.3 When the Timing of the ENCAnEC Input is the Same as that of the ENCAnE1 Input during Up-count (When ENCAnACL = 1, ENCAnBCL = 0, ENCAnZCL = 1, and ENCAnUDS[1:0] = $11_B$) ..... 2403
28.6.18.4 When the Timing of the ENCAnEC Input is Earlier than that of the ENCAnE1 Input during Up-count (When ENCAnACL = 1, ENCAnBCL = 0, ENCAnZCL = 1, and ENCAnUDS[1:0] = $11_B$) ..... 2403
28.6.18.5 When the Timing of the ENCAnEC Input is Later than that of the ENCAnE1 Input during Down-count (When ENCAnACL = 1, ENCAnBCL = 0, ENCAnZCL = 1, and ENCAnUDS[1:0] = $11_B$) ..... 2404
28.6.19 Capture Operation Performed upon Clearing by ENCAnEC when ENCAnSCE = 0 ..... 2405

Section 29 Motor Control ..... 2406

29.1 Features of RH850/F1K Motor Control ..... 2406
29.1.1 Number of Units and Channels ..... 2406
29.1.2 Register Base Address ..... 2407
29.1.3 Clock Supply ..... 2407
29.1.4 Interrupt Request ..... 2407
29.1.5 Reset Sources ..... 2407
29.1.6 External Input/Output Signal ..... 2408
29.1.7 Internal Output Signal ..... 2409
29.2 Overview ..... 2410
29.2.1 Functional Overview ..... 2410
29.2.2 Basic Structure of Motor Control ..... 2411
29.2.3 Block Diagram ..... 2412
29.2.4 Definition of Terms ..... 2413
29.3 Registers ..... 2414
29.3.1 List of Registers ..... 2414
29.3.2 TAPAnCTL0 — TAPA Control Register 0 ..... 2415
29.3.3 TAPAnCTL1 — TAPA Control Register 1 ..... 2416
29.3.4 TAPAnFLG — TAPA Flag Register ..... 2417
29.3.5 TAPAnACWE — TAPA Asynchronous Hi-Z Control Write Enable Register ..... 2418

29.3.6 TAPAnACTS — TAPA Asynchronous Hi-Z Control Start Trigger Register ..... 2418
29.3.7 TAPAnACTT — TAPA Asynchronous Hi-Z Control Stop Trigger Register ..... 2419
29.3.8 TAPAnOPHS — TAPA Hi-Z Start Trigger Register ..... 2419
29.3.9 TAPAnOPHT — TAPA Hi-Z Stop Trigger Register ..... 2420
29.3.10 TAPAnEMU — TAPA Emulation Register ..... 2420
29.4 Asynchronous Hi-Z Control Function ..... 2421
29.4.1 Overview ..... 2421
29.4.2 System Configuration Example ..... 2421
29.4.3 Basic Operation ..... 2423
29.4.4 Asynchronous Hi-Z Control Using Software Trigger ..... 2425
29.4.5 Operating Procedure ..... 2426
29.4.6 TAPA0 Hi-Z Control Input Selection ..... 2427
29.4.7 Registers ..... 2428
29.4.7.1 PIC0HIZCENn — Hi-Z Output Control Register n (n = 0) ..... 2428
29.5 INT Signal Output Selection Function ..... 2429
29.5.1 Configuration of the INT Signal Output Selection Function ..... 2429
29.5.2 Block Diagram ..... 2430
29.5.3 Registers ..... 2431
29.5.3.1 PIC0REG2n0 — Timer I/O Control Register 2n0 (n = 0) ..... 2431
29.6 A/D Converter Conversion Trigger Selection Function ..... 2432
29.6.1 Configuration of A/D Converter Conversion Trigger Selection Function ..... 2432
29.6.2 Block Diagram ..... 2433
29.6.3 Waveforms of A/D Converter Conversion Trigger Output Control Operation in Triangle PWM Mode ..... 2434
29.6.4 Operating Procedure for A/D Converter Conversion Trigger Selection Function ..... 2436
29.7 ADCA Trigger Selection Function ..... 2437
29.7.1 Functional Overview ..... 2437
29.7.2 Configuration ..... 2437
29.7.3 Registers ..... 2438
29.7.3.1 PIC0ADTEN4nj — A/D Conversion Trigger Output Control Register 4nj (n = 0, j = 0 to 2) ..... 2438
29.7.4 Example of Operation ..... 2439
29.7.5 Setup Flow ..... 2439
29.8 Simultaneous Start Trigger Function ..... 2440
29.8.1 Functional Overview ..... 2440
29.8.2 Configuration ..... 2440
29.8.3 Registers ..... 2441
29.8.3.1 PIC0SSER0 — Simultaneous Start Control Register 0 ..... 2441
29.8.3.2 PIC0SSER2 — Simultaneous Start Control Register 2 ..... 2441
29.8.3.3 PIC0SST — Simultaneous Start Trigger Control Register ..... 2442
29.8.4 Example of Operation ..... 2443
29.8.5 Setup Flow ..... 2444
29.9 Trigger & Pulse Width Measuring Function ..... 2445
29.9.1 Functional Overview ..... 2445
29.9.2 Configuration ..... 2445
29.9.3 Registers ..... 2447

29.9.3.1 PIC0REG31 — Timer I/O Control Register 31 ... 2447
29.9.4 Example of Operation ... 2449
29.9.5 Setup Flow ... 2451
29.9.6 Setting Examples for Operation Functions ... 2454
29.10 A/D Trigger Encoder Capture Function ... 2457
29.10.1 Functional Overview ... 2457
29.10.2 Configuration ... 2457
29.10.3 Registers ... 2458
29.10.3.1 PIC0REG30 — Timer I/O Control Register 30 ... 2458
29.10.4 Example of Operation ... 2459
29.10.5 Setup Flow ... 2460
29.10.6 Setting Examples for Operation Functions ... 2461
29.11 Three-Phase PWM Output with Dead Time ... 2462
29.11.1 Functional Overview ... 2462
29.11.2 Configuration ... 2462
29.11.3 Registers ... 2465
29.11.3.1 PIC0REG2n2 — Timer I/O Control Register 2n2 (n = 0) ... 2465
29.11.3.2 PIC0HIZCENn — Hi-Z Output Control Register n (n = 0) ... 2466
29.11.4 Operation Example ... 2467
29.11.4.1 Pwm Output ... 2467
29.11.4.2 One-Phase PWM Output ... 2467
29.11.4.3 SR Flip-Flop Circuit ... 2468
29.11.5 Setup Flow ... 2476
29.11.6 Setting Examples for Operation Functions ... 2478
29.11.6.1 TAUDn Settings (Active High Example) ... 2478
29.11.6.2 PIC Settings ... 2483
29.12 High-accuracy Triangle PWM Output with Dead Time ... 2484
29.12.1 Functional Overview ... 2484
29.12.2 Configuration ... 2485
29.12.3 Registers ... 2487
29.12.3.1 PIC0REG2n0 — Timer I/O Control Register 2n0 (n = 0) ... 2487
29.12.3.2 PIC0REG2n1 — Timer I/O Control Register 2n1 (n = 0) ... 2488
29.12.3.3 PIC0REG2n2 — Timer I/O Control Register 2n2 (n = 0) ... 2490
29.12.3.4 PIC0REG2n3 — Timer I/O Control Register 2n3 (n = 0) ... 2491
29.12.3.5 PIC0HIZCENn — Hi-Z Output Control Register n (n = 0) ... 2493
29.12.4 Operation Example ... 2494
29.12.4.1 Triangle PWM Output with Dead Time ... 2494
29.12.4.2 One-shot Pulse Output ... 2494
29.12.4.3 U phase Combination Circuit (PFN001) ... 2495
29.12.4.4 Logical Operation Circuit (FN0i) (i = 0 or 1) ... 2497
29.12.5 Setup Flow ... 2507
29.12.6 Setting Examples for Operation Functions ... 2509
29.12.6.1 TAUDn settings (active high example) ... 2509
29.12.6.2 PIC Settings (Active High Example) ... 2515
29.13 Delay Pulse Output with Dead Time ... 2516
29.13.1 Functional Overview ... 2516
29.13.2 Configuration ... 2516

29.13.2.1 TAUDn configuration ..... 2517
29.13.3 Registers ..... 2519
29.13.3.1 PIC0REG2n2 — Timer I/O Control Register 2n2 (n = 0) ..... 2519
29.13.3.2 PIC0HIZCENn — Hi-Z Output Control Register n (n = 0) ..... 2520
29.13.4 Operation Example ..... 2521
29.13.4.1 Delay pulse output function ..... 2521
29.13.4.2 One-phase PWM output ..... 2521
29.13.5 Setup Flow ..... 2527
29.13.6 Setting Examples for Operation Functions ..... 2530
29.13.6.1 TAUDn Settings ..... 2530
29.13.6.2 Peripheral Interconnections Settings ..... 2535

Section 30 PWM Output/Diagnostic (PWM-Diag) ..... 2536

30.1 Features of RH850/F1K PWM-Diag ..... 2536
30.1.1 Number of Units and Channels ..... 2536
30.1.2 Register Base Address ..... 2537
30.1.3 Clock Supply ..... 2537
30.1.4 Interrupt Requests ..... 2538
30.1.5 Reset Sources ..... 2540
30.1.6 External Input/Output Signals ..... 2540
30.1.7 Internal Signals ..... 2540
30.1.8 Functional Overview ..... 2541
30.1.9 Block Diagram ..... 2543
30.2 Registers ..... 2544
30.2.1 List of Registers ..... 2544
30.2.1.1 PWBAnBRSm Register ..... 2545
30.2.1.2 PWBAnTE Register ..... 2546
30.2.1.3 PWBAnTS Register ..... 2547
30.2.1.4 PWBAnTT Register ..... 2548
30.2.1.5 PWBAnEMU Register ..... 2549
30.2.1.6 PWGAnCTL — PWGA Control Register ..... 2550
30.2.1.7 PWGAnCNT — PWM Cycle Count Register ..... 2550
30.2.1.8 PWGAnCSDR — PWM Output Set Condition Register ..... 2551
30.2.1.9 PWGAnCRDR — PWM Output Reset Condition Register ..... 2551
30.2.1.10 PWGAnCTDR — PWGA_TRGOUTn Generation Condition Register ..... 2552
30.2.1.11 PWGAnCSBR — PWGAnCSDR Buffer Register ..... 2552
30.2.1.12 PWGAnCRBR — PWGAnCRDR Buffer Register ..... 2553
30.2.1.13 PWGAnCTBR — PWGAnCTDR Buffer Register ..... 2553
30.2.1.14 PWGAnRSF — Buffer Register Reload Status Register ..... 2554
30.2.1.15 PWGAnRDT — Buffer Register Reload Trigger Register ..... 2554
30.2.1.16 SLPWGAk — PWGA Synchronous Trigger Register (k = 0 to 2) ..... 2555
30.2.1.17 PWSAnCTL Register ..... 2556
30.2.1.18 PWSAnSTR Register ..... 2556
30.2.1.19 PWSAnSTC Register ..... 2557
30.2.1.20 PWSAnQUEj (j = 0 to 7) Register ..... 2557
30.2.1.21 PWSAnPVCRx_y (x = 00, 02, 04 ... 70, y = 01, 03, 05 ... 71) Register ..... 2558
30.2.1.22 PWSAnEMU — Emulation Control Register ..... 2559
30.3 Operating Procedure ..... 2560

30.4 Operation Waveform of PWM-Diag ..... 2562
30.4.1 PWM Waveform Output by PWGA and Operation Waveform for A/D Conversion Trigger Output ..... 2562
30.4.1.1 Basic Operation Waveform of PWGA ..... 2562
30.4.1.2 Operation Waveform when Simultaneous Rewrite for PWGA is Executed ..... 2563
30.4.1.3 Operation Waveform when Stopping and Restarting PWGA Operation ..... 2564
30.4.1.4 Waveforms of PWGA Operation with Specific Settings ..... 2566
30.4.2 Operation Waveform when A/D Conversion Trigger Occurs in PWSA ..... 2567
30.5 PWM-Diag Related Function in A/D Converter (ADCA) ..... 2568
30.5.1 ADCA Registers when the PWM-Diag Function is Used ..... 2568

Section 31 A/D Converter (ADCA) ..... 2569

31.1 Features of RH850/F1K ADCA ..... 2569
31.1.1 Number of Units and Channels ..... 2569
31.1.2 Register Base Address ..... 2570
31.1.3 Clock Supply ..... 2571
31.1.4 Interrupt Requests ..... 2571
31.1.5 Reset Sources ..... 2571
31.1.6 External Input/Output Signals ..... 2572
31.2 Overview ..... 2575
31.2.1 Functional Overview ..... 2575
31.2.2 Block Diagram ..... 2577
31.3 Registers ..... 2582
31.3.1 List of Registers ..... 2582
31.3.2 ADCA Specific Registers ..... 2584
31.3.2.1 ADCAnVCRj — Virtual Channel Register j ..... 2584
31.3.2.2 ADCAnPWDVCR — PWM-Diag Virtual Channel Register ..... 2586
31.3.2.3 ADCAnDRj — Data Register j ..... 2587
31.3.2.4 ADCAnDIRj — Data Supplementary Information Register j ..... 2589
31.3.2.5 ADCAnPWDTSNDR — PWM-Diag Data Register ..... 2590
31.3.2.6 ADCAnPWDDIR — PWM-Diag Data Supplementary Information Register ..... 2591
31.3.2.7 ADCAnADHALTR — A/D Force Halt Register ..... 2592
31.3.2.8 ADCAnADCR — A/D Control Register ..... 2593
31.3.2.9 ADCAnMPXCURR — MPX Current Register ..... 2595
31.3.2.10 ADCAnTHSMPSTCR — T&H Sampling Start Control Register ..... 2596
31.3.2.11 ADCAnTHCR — T&H Control Register ..... 2597
31.3.2.12 ADCAnTHAHLDSTCR — T&H Group A Hold Start Control Register ..... 2598
31.3.2.13 ADCAnTHBHLDSTCR — T&H Group B Hold Start Control Register ..... 2599
31.3.2.14 ADCAnTHACR — T&H Group A Control Register ..... 2600
31.3.2.15 ADCAnTHBCR — T&H Group B Control Register ..... 2601
31.3.2.16 ADCAnTHER — T&H Enable Register ..... 2603
31.3.2.17 ADCAnTHGSR — T&H Group Select Register ..... 2604
31.3.2.18 ADCAnSMPCR — Sampling Control Register ..... 2606
31.3.2.19 ADCAnSFTCR — Safety Control Register ..... 2607
31.3.2.20 ADCAnULLMTBR0 to 2 — Upper Limit/Lower Limit Table Registers 0 to 2 ..... 2608
31.3.2.21 ADCAnECR — Error Clear Register ..... 2609
31.3.2.22 ADCAnULER — Upper Limit/Lower Limit Error Register ..... 2610

31.3.2.23 ADCAnOWER — Overwrite Error Register ... 2612
31.3.3 Scan Group (SG) Specific Registers ... 2613
31.3.3.1 ADCAnSGSTCRx — Scan Group x Start Control Register ... 2613
31.3.3.2 ADCAnSGCRx — Scan Group x Control Register ... 2614
31.3.3.3 ADCAnPWDSGCR — PWM-Diag Scan Group Control Register ... 2615
31.3.3.4 ADCAnSGVCSPx — Scan Group x Start Virtual Channel Pointer ... 2616
31.3.3.5 ADCAnSGVCEPx — Scan Group x End Virtual Channel Pointer ... 2617
31.3.3.6 ADCAnSGMCYCRx — Scan Group x Multicycle Register ... 2618
31.3.3.7 ADCAnPWDSGSEFCR — PWM-Diag Scan End Flag Clear Register ... 2619
31.3.3.8 ADCAnSGSEFCRx — Scan Group x Scan End Flag Clear Register ... 2620
31.3.3.9 ADCAnSGSTR — Scan Group Status Register ... 2621
31.3.4 Hardware Trigger Specific Register ... 2623
31.3.4.1 ADCAnSGTSELx — Scan Group x Start Trigger Control Register x ... 2623
31.3.5 Self-Diagnosis Specific Registers ... 2625
31.3.5.1 ADCAnDGCTL0 — Self-Diagnosis Control Register 0 ... 2625
31.3.5.2 ADCAnDGCTL1 — Self-Diagnosis Control Register 1 ... 2626
31.3.5.3 ADCAnPDCTL1 — Pull Down Control Register 1 ... 2627
31.3.5.4 ADCAnPDCTL2 — Pull Down Control Register 2 ... 2628
31.3.6 Emulation Specific Register ... 2629
31.3.6.1 ADCAnEMU — Emulation Control Register ... 2629
31.4 Operation ... 2630
31.4.1 Initial Setting ... 2630
31.4.2 Trigger Input ... 2631
31.4.3 Ending A/D Conversion ... 2632
31.4.4 Example of Scan Group Operation ... 2633
31.4.5 Channel Repeat Mode ... 2635
31.4.6 Example of Simultaneous Track and Hold Operation ... 2637
31.4.7 A/D Conversion with External Analog Multiplexer ... 2639
31.4.7.1 A/D Conversion with PWM-Diag Enabled ... 2641
31.4.8 Example of Synchronous Suspend and Resume Operation ... 2643
31.4.9 Example of Asynchronous Suspend and Resume Operation ... 2644
31.4.10 Error Detecting Functions ... 2645
31.4.10.1 Upper-Limit/Lower-Limit Error Detecting Function ... 2645
31.4.10.2 Overwrite Error Detecting Function ... 2645
31.4.10.3 SVSTOP Operation ... 2645
31.4.11 Activating Scan Group by a Hardware Trigger ... 2649
31.4.11.1 Stopping Scan Group by ADHALT ... 2649
31.4.12 Scan End Interrupt Request ... 2650
31.4.13 A/D Error Interrupt Request ... 2651
31.5 Self-Diagnostic Function ... 2652
31.5.1 Diagnosis of A/D Conversion Circuit ... 2653
31.5.1.1 Diagnostic procedure ... 2653
31.5.2 Diagnosis of Channel Multiplexer ... 2654
31.5.2.1 Diagnostic procedure ... 2655
31.5.3 Diagnosis of Open Pins ... 2656
31.5.3.1 Diagnostic procedure ... 2657
31.5.4 Diagnosis of T&H Circuit ... 2658

31.5.4.1 Diagnostic Procedure (in case of T&H circuit ch0 diagnosis) ... 2658
31.5.4.2 Diagnosis Mechanism ... 2660
31.6 Definition of A/D Conversion Accuracy ... 2661
31.7 Usage Notes ... 2662
31.7.1 Range of Channel Input Voltage ... 2662
31.7.2 Notes on Application Design ... 2662

Section 32 Key Return (KR) ... 2667

32.1 Features of RH850/F1K KR ... 2667
32.1.1 Number of Units and Channels ... 2667
32.1.2 Register Base Address ... 2667
32.1.3 Clock Supply ... 2668
32.1.4 Interrupt Requests ... 2668
32.1.5 Reset Sources ... 2668
32.1.6 External Input/Output Signals ... 2668
32.2 Overview ... 2669
32.2.1 Functional Overview ... 2669
32.2.2 Block Diagram ... 2669
32.3 Registers ... 2670
32.3.1 List of Registers ... 2670
32.3.2 KRnKRM — Key Return Mode Register ... 2670
32.4 Operation ... 2671
32.4.1 Interrupt Request INTKRn ... 2671

Section 33 Functional Safety ... 2672

33.1 Overview ... 2672
33.2 ECC ... 2673
33.2.1 Overview ... 2673
33.2.2 Code Flash ECC ... 2675
33.2.2.1 Overview ... 2675
33.2.2.2 Interrupt Requests ... 2676
33.2.2.3 List of Registers ... 2677
33.2.2.4 Details of Registers ... 2678
33.2.3 Data Flash ECC ... 2686
33.2.3.1 Overview ... 2686
33.2.3.2 Interrupt Requests ... 2686
33.2.3.3 List of Registers ... 2687
33.2.3.4 Details of Registers ... 2688
33.2.4 Local RAM (Including the Retention RAM) ECC ... 2694
33.2.4.1 Overview ... 2694
33.2.4.2 Interrupt Requests ... 2695
33.2.4.3 List of Registers ... 2696
33.2.4.4 Details of Registers ... 2697
33.2.5 Peripheral RAM ECC ... 2704
33.3 Memory Protection ... 2705

33.3.1 Overview ... 2705
33.3.1.1 Identifiers for Slave Guard ... 2706
33.3.2 PBG ... 2707
33.3.2.1 List of Registers ... 2711
33.3.2.2 Details of Registers ... 2716
33.3.3 PBG for CPU System ... 2722
33.3.3.1 List of Registers ... 2723
33.3.3.2 Details of Registers ... 2724

Section 34 Data CRC (DCRA) ... 2730

34.1 Features of RH850/F1K DCRA ... 2730
34.1.1 Number of Units ... 2730
34.1.2 Register Base Address ... 2730
34.1.3 Clock Supply ... 2731
34.1.4 Reset Sources ... 2731
34.2 Overview ... 2732
34.2.1 Functional Overview ... 2732
34.2.2 Block Diagram ... 2732
34.2.3 Operational Circuit ... 2733
34.3 Registers ... 2734
34.3.1 List of Registers ... 2734
34.3.2 DCRAnCIN — CRC Input Register ... 2735
34.3.3 DCRAnCOUT — CRC Data Register ... 2736
34.3.4 DCRAnCTL — CRC Control Register ... 2737
34.4 Operation ... 2738

Section 35 Security Function ... 2739

Section 36 On-Chip Debug Unit (OCD) ... 2740

36.1 Overview of RH850/F1K OCD ... 2740
36.1.1 Functional Overview ... 2740
36.1.2 External Input/Output Pins ... 2742
36.2 Peripheral Break Control ... 2743
36.3 Hot Plug-in in Each Mode ... 2745
36.3.1 RUN Mode ... 2745
36.3.2 STOP/DeepSTOP Mode ... 2745
36.3.3 Cyclic RUN Mode ... 2745
36.3.4 Cyclic STOP Mode ... 2746
36.4 Registers ... 2747
36.4.1 EPC — Emulation Peripheral Control Register ... 2747
36.5 Cautions on Using On-Chip Debugging ... 2748
36.5.1 Treatment of Devices Used for Debugging ... 2748
36.5.2 Reset Assertion When a Debugger is Connected ... 2748
36.5.3 Restrictions When HS IntOSC is Used as the Main Clock Source Instead of MainOSC ... 2748

36.5.4 Restrictions When the Writing of OCD_MD and RESET are Occur at the Same Time, or Restrictions When the Writing of MTR (DBG_CTRLP) and RESET are Occur at the Same Time ..... 2748
36.5.5 Transition to DeepSTOP Mode When a Debugger is Connected ..... 2748

Section 37 Flash Memory ..... 2749

37.1 Features ..... 2749
37.2 Structure of Memory ..... 2750
37.2.1 Mapping of Code Flash Memory ..... 2750
37.2.2 Mapping of Data Flash Memory ..... 2750
37.3 Operating Modes Associated with Flash Memory ..... 2751
37.4 Functions ..... 2752
37.4.1 Functional Overview ..... 2752
37.5 Serial Programming ..... 2756
37.5.1 Environments for Programming ..... 2756
37.6 Communication Modes ..... 2757
37.6.1 Asynchronous Flash Programming Interface - 1-Wire UART ..... 2757
37.6.2 Asynchronous Flash Programming Interface - 2-Wire UART ..... 2757
37.6.3 Synchronous Flash Programming Interface CSI ..... 2757
37.6.4 Selection of Communication Method ..... 2758
37.7 Self-Programming ..... 2759
37.7.1 Outline ..... 2759
37.7.2 Background Operation ..... 2760
37.7.3 Enabling Self-Programming ..... 2760
37.7.3.1 FLMDCNT Register ..... 2760
37.8 Reading Flash Memory ..... 2761
37.8.1 Reading Code Flash Memory ..... 2761
37.8.2 Reading Data Flash Memory ..... 2761
37.8.2.1 EEPRDCYCL — Data Flash Wait Cycle Control Register ..... 2762
37.8.2.2 PRDNAMEn — Product Name Storage Register (n = 1 to 3) ..... 2763
37.8.2.3 CHIPIDnXX — Chip ID Register (n = 1,2, XX = LL,LH,HL,HH) ..... 2765
37.9 Option Bytes ..... 2766
37.9.1 Option Byte Setting ..... 2766
37.9.2 OPBT0 — Option Byte 0 ..... 2767
37.9.3 OPBT1 — Option Byte 1 ..... 2769
37.10 Usage Notes ..... 2770

Section 38 RAM ..... 2772

38.1 Features ..... 2772
38.2 Memory Configuration ..... 2773
38.3 Usage Notes ..... 2774

Section 39 Boundary Scan ..... 2775

39.1 Overview ..... 2775

39.2 Features ... 2775
39.3 External Input/Output Pins ... 2777
39.4 Register Descriptions ... 2778
39.4.1 Instruction Register (SDIR) ... 2779
39.4.2 ID Register (SDID) ... 2779
39.4.3 Bypass Register (SDBPR) ... 2779
39.4.4 Boundary Scan Register (SDBSR) ... 2779
39.5 Operation ... 2780
39.5.1 TAP Controller ... 2780
39.5.2 Supported Instructions ... 2781
39.5.2.1 BYPASS ... 2781
39.5.2.2 SAMPLE/PRELOAD ... 2781
39.5.2.3 EXTEST ... 2781
39.5.2.4 IDCODE ... 2781
39.5.3 Pins Subject to Boundary Scan ... 2782
39.6 Usage Notes ... 2783

Section 40 Electrical Characteristics ... 2784

40.1 Overview ... 2784
40.1.1 Pin Groups ... 2784
40.1.1.1 176 pin ... 2784
40.1.1.2 144 pin ... 2784
40.1.1.3 100 pin ... 2784
40.1.2 General Measurement Conditions ... 2784
40.1.2.1 Common Conditions ... 2784
40.1.2.2 AC Characteristic Measurement Condition ... 2785
40.2 Absolute Maximum Ratings ... 2786
40.2.1 Supply Voltages ... 2786
40.2.2 Port Voltages ... 2786
40.2.3 Port Current ... 2787
40.2.3.1 176 pin ... 2788
40.2.3.2 144 pin ... 2789
40.2.3.3 100 pin ... 2790
40.2.4 Temperature Condition ... 2790
40.3 Capacitance ... 2791
40.4 Operational Condition ... 2792
40.5 Oscillator Characteristics ... 2794
40.6 Internal Oscillator Characteristics ... 2796
40.7 PLL Characteristics ... 2797
40.8 Power Management Characteristics ... 2798
40.8.1 Regulator Characteristics ... 2798
40.8.2 Voltage Detector (POC, LVI, VLVI, CVM) Characteristics ... 2799
40.8.3 Power Up/Down Timing ... 2802
40.8.4 CPU Reset Release Timing ... 2807
40.9 Pin Characteristics ... 2808

40.9.1 Output Current ... 2814
40.9.1.1 176 pin ... 2814
40.9.1.2 144 pin ... 2815
40.9.1.3 100 pin ... 2816
40.10 Power Supply Currents ... 2817
40.11 Interrupt Timing ... 2819
40.12 $\overline{\text{RESET}}$ Timing ... 2820
40.13 Low Power Sampler (DPIN input) Timing ... 2820
40.14 CSCXFOUT Timing ... 2821
40.15 Mode Timing ... 2822
40.16 Timer Timing ... 2823
40.17 RLIN2/RLIN3 timing ... 2825
40.18 RS-CAN Timing ... 2825
40.19 CSI Timing ... 2826
40.19.1 CSIG Timing ... 2826
40.19.2 CSIH Timing ... 2827
40.20 RIIC Timing ... 2837
40.21 ADTRG Timing ... 2840
40.22 Key Return Timing ... 2840
40.23 $\overline{\text{DCUTRST}}$ Timing ... 2841
40.24 Debug Interface Characteristics ... 2842
40.24.1 Nexus Interface Timing ... 2842
40.24.2 LPD (4 pin) Interface Timing ... 2843
40.24.3 LPD (1 pin) Interface Timing ... 2844
40.25 Flash Programming Characteristics ... 2845
40.25.1 Code Flash ... 2845
40.25.2 Data Flash ... 2847
40.25.3 Serial Programming Interface ... 2848
40.25.3.1 Serial Programmer Setup Timing ... 2848
40.25.3.2 Flash Programming Interface ... 2849
40.26 A/D Converter Characteristics ... 2850
40.26.1 Equivalent Circuit of the Analog Input Block ... 2852
40.27 Injection Currents ... 2853
40.27.1 Absolute Maximum Ratings ... 2854
40.27.1.1 176 pin and 144 pin ... 2854
40.27.1.2 100 pin ... 2855
40.27.2 DC Characteristics for Overload Current ... 2856
40.27.2.1 176 pin and 144 pin ... 2856
40.27.2.2 100 pin ... 2856
40.28 Thermal Characteristics ... 2857
40.28.1 Parameters ... 2857
40.28.2 Board ... 2857

Appendix A. Package ..... 2858
A.1 Package Dimensions ..... 2858
A.1.1 176 Pin ..... 2858
A.1.2 144 Pin ..... 2859
A.1.3 100 Pin ..... 2860

# Section 1 Overview

## 1.1 RH850/F1K Product Features

The features of the RH850/F1K are described below.

The RH850/F1K is a 32-bit single-chip microcontroller with a G3KH CPU core. The key features of the F1K are low power consumption, high computational processing power, and a wide variety of internal peripheral functions. To reduce supply current in a variety of applications, a wide range of power reducing measures are available. For example, there is a Low Power Sampler (LPS), that can poll signals input to the analog and digital input pins without CPU core interaction, and DeepSTOP mode in which the power supply to the most circuits of the microcontroller can be turned off.

### Applications

The RH850/F1K is ideal for automotive electronics, such as BCM (body control module), gateway, HVAC, lighting modules, and many other applications.

## 1.2 RH850/F1K Functions

Table 1.1 Overview of Product (1/2)

| Product name | | | RH850/F1K | | |
|---|---|---|---|---|---|
| | | | 100 pin | 144 pin | 176 pin |
| Memory | | | See **Table 1.2, Product Lineup**. | | |
| External memory interface (MEMC) | | | Not provided | | |
| CPU | CPU System | | G3KH | | |
| | CPU frequency | | 120 MHz max. (ADVANCED/ PREMIUM) | 80 MHz max. (ECO)<br>120 MHz max. (ADVANCED/PREMIUM) | |
| | FPU | | Single-precision | | |
| | Memory Protection Unit (MPU) | | Provided | | |
| DMA | | | 16 channels | | |
| Operating clock | Main Oscillator (MainOSC) | | 16/20/24 MHz | | |
| | Low Speed Internal Oscillator (LS IntOSC) | | 240 kHz (typ.) | | |
| | High Speed Internal Oscillator (HS IntOSC) | | 8 MHz (typ.) | | |
| | PLL | | Provided | | |
| | Sub Oscillator (SubOSC) | | Not provided | 32.768 kHz | |
| I/O port | | | 81 | 120 | 150 |
| A/D converter | ADC0 | Physical input channels | Total 36 ch (12 bit resolution: 16 ch + 10 bit resolution: 20 ch) | Total 36 ch (12 bit resolution: 16 ch + 10 bit resolution: 20 ch) | Total 36 ch (12 bit resolution: 16 ch + 10 bit resolution: 20 ch) |
| | | External multiplexer support for channel number extension | Provided | | |
| | | Channels with T&H | 6 | | |
| | ADC1 | Physical input channels | Not provided | Total 12 ch (12 bit resolution: 8 ch + 10 bit resolution: 4 ch) | Total 24 ch (12 bit resolution: 16 ch + 10 bit resolution: 8 ch) |
| | | External multiplexer support for channel number extension | Not provided | | |
| | | Channels with T&H | Not provided | | |
| Timer | Timer Array Unit D (TAUD) | | 1 unit (16 bit resolution timers × 16 channels /unit) | | |
| | Timer Array Unit B (TAUB) | | 1 unit (16 bit resolution timers × 16 channels /unit) | | 2 units (16 bit resolution timers × 16 channels /unit) |
| | Timer Array Unit J (TAUJ) | | 2 units (32 bit resolution timers × 4 channels /unit) | | |
| | Operating System Timer (OSTM) | | 5 units | | |
| | Real-Time Counter (RTCA) | | Not provided | 1 unit | |
| | Encoder Timer (ENCA) | | 1 unit | | |
| | Window Watchdog Timer A (WDTA) | | 2 units | | |

**Table 1.1 Overview of Product (2/2)**

| Product name | | RH850/F1K | | |
|---|---|---|---|---|
| | | 100 pin | 144 pin | 176 pin |
| Serial interfaces | Clocked Serial Interface G (CSIG) | 1 channel | 2 channels | |
| | Clocked Serial Interface H (CSIH) | 4 channels | | |
| | CAN Interface | ADVANCED | ECO/ADVANCED | |
| | | RSCAN<br>6 channels<br>(Total 480 message buffers) | RSCAN<br>6 channels<br>(Total 480 message buffers) | RSCAN<br>7 channels<br>(Total 560 message buffers) |
| | | PREMIUM | | |
| | | RS-CANFD<br>6 channels<br>(Total 480 message buffers) | RS-CANFD<br>6 channels<br>(Total 480 message buffers) | RS-CANFD<br>6 channels<br>+<br>RSCAN<br>1 channel<br>(Total 560 message buffers) |
| | LIN/UART Interface (RLIN3) | 4 channels | 6 channels | |
| | LIN Master Interface (RLIN2) | 3 channels | 6 channels | 10 channels |
| | $I^2C$ Interface (RIIC) | 1 channel | | |
| External Interrupts | Maskable | 13 | 16 | |
| | Non-maskable (NMI) | 1 | | |
| Other functions | Clock Monitors (CLMA) | For PLL, HS IntOSC, MainOSC | | |
| | Data CRC (DCRA) | 4 channels | | |
| | Low-Voltage Indicator (LVI) | Provided | | |
| | Power-On-Clear (POC) | Provided | | |
| | Core Voltage Monitors (CVM) | Provided | | |
| | Error Correction Coding (ECC) | For Code flash, Data flash, Local RAM, Retention RAM, CSIH, RS-CAN | | |
| | Low Power Sampling (LPS) | Provided | | |
| | PWM diagnosis (PWM_DIAG) | 48 channels | 64 channels | 72 channels |
| | Motor Control | 1 unit | | |
| | Key Return (KR) | 8 channels | | |
| | CLOCK OUTPUT (FOUT) | Provided | | |
| | RESET OUTPUT (RESETOUT) | Provided | | |
| | ICUSE (Intelligent Cryptographic Unit E) | Provided | | |
| | SWDT (Secure WDT) | Provided | | |
| | On-Chip debug (OCD) | Provided | | |
| | Boundary Scan | Provided | | |
| Voltage supply | Internal supply | VPOC to 5.5 V | | |
| | Input/output buffer supplies | VPOC to 5.5 V | | |
| | A/D Converter supplies | 3.0 to 5.5 V | | |
| Package | | 100 pin LQFP | 144 pin LQFP | 176pin LQFP |

## 1.3 RH850/F1K Product Lineup

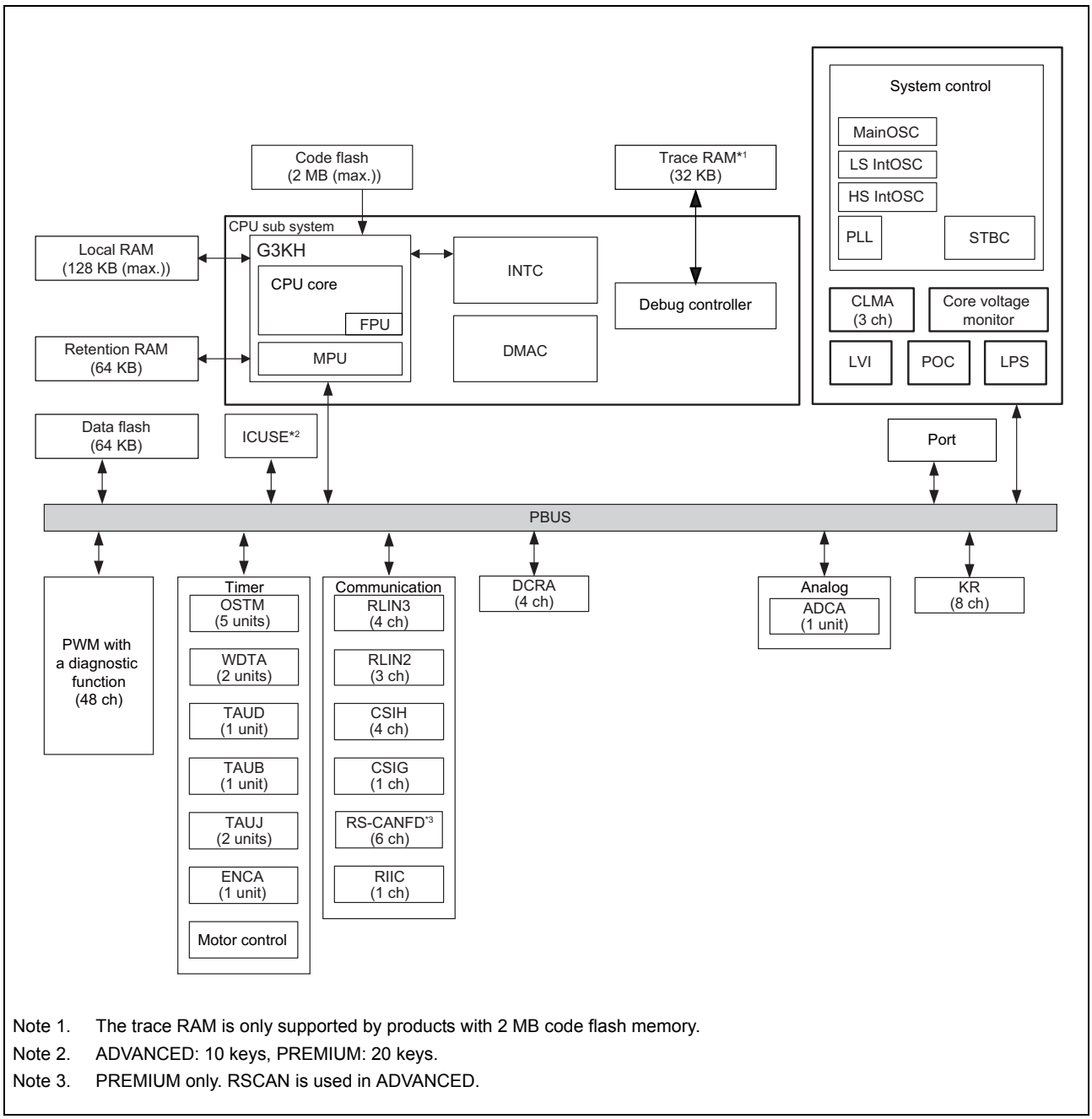
Table 1.2 Product Lineup (1/2)

| Pin Count | CPU frequency | Memory | | | | CAN Interface (RSCAN) | CAN FD Interface (RS-CANFD) | Part Name | | Line Name |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Code Flash | Local RAM | Data Flash | Retention RAM (RRAM) | | | Operating Temperature (Ta) | | |
| | | | | | | | | –40°C to +105°C | –40°C to +125°C | |
| 144 pins | 80 MHz max. | 768 KB | 32 KB | 64 KB | 64 KB | 6 channels | — | R7F7016023AFP<br>R7F7016023AFP-C | R7F7016024AFP<br>R7F7016024AFP-C | ECO |
| | | 1024 KB | 64 KB | | | | | R7F7016033AFP<br>R7F7016033AFP-C | R7F7016034AFP<br>R7F7016034AFP-C | |
| | | 1536 KB | 96 KB | | | | | R7F7015423AFP<br>R7F7015423AFP-C | R7F7015424AFP<br>R7F7015424AFP-C | |
| | | 2048 KB | 128 KB | | | | | R7F7015433AFP<br>R7F7015433AFP-C | R7F7015434AFP<br>R7F7015434AFP-C | |
| 176 pins | 80 MHz max. | 1024 KB | 64 KB | 64 KB | 64 KB | 7 channels | — | R7F7015573AFP<br>R7F7015573AFP-C | R7F7015574AFP<br>R7F7015574AFP-C | |
| | | 1536 KB | 96 KB | | | | | R7F7015463AFP<br>R7F7015463AFP-C | R7F7015464AFP<br>R7F7015464AFP-C | |
| | | 2048 KB | 128 KB | | | | | R7F7015473AFP<br>R7F7015473AFP-C | R7F7015474AFP<br>R7F7015474AFP-C | |
| 100 pins | 120 MHz max. | 768 KB | 32 KB | 64 KB | 64 KB | 6 channels | — | R7F7016103AFP<br>R7F7016103AFP-C | R7F7016104AFP<br>R7F7016104AFP-C | ADVANCED |
| | | 1024 KB | 64 KB | | | | | R7F7016113AFP<br>R7F7016113AFP-C | R7F7016114AFP<br>R7F7016114AFP-C | |
| | | 1536 KB | 96 KB | | | | | R7F7015603AFP<br>R7F7015603AFP-C | R7F7015604AFP<br>R7F7015604AFP-C | |
| | | 2048 KB | 128 KB | | | | | R7F7015613AFP<br>R7F7015613AFP-C | R7F7015614AFP<br>R7F7015614AFP-C | |
| 144 pins | 120 MHz max. | 768 KB | 32 KB | 64 KB | 64 KB | 6 channels | — | R7F7016123AFP<br>R7F7016123AFP-C | R7F7016124AFP<br>R7F7016124AFP-C | |
| | | 1024 KB | 64 KB | | | | | R7F7016133AFP<br>R7F7016133AFP-C | R7F7016134AFP<br>R7F7016134AFP-C | |
| | | 1536 KB | 96 KB | | | | | R7F7015623AFP<br>R7F7015623AFP-C | R7F7015624AFP<br>R7F7015624AFP-C | |
| | | 2048 KB | 128 KB | | | | | R7F7015633AFP<br>R7F7015633AFP-C | R7F7015634AFP<br>R7F7015634AFP-C | |

Table 1.2 Product Lineup (2/2)

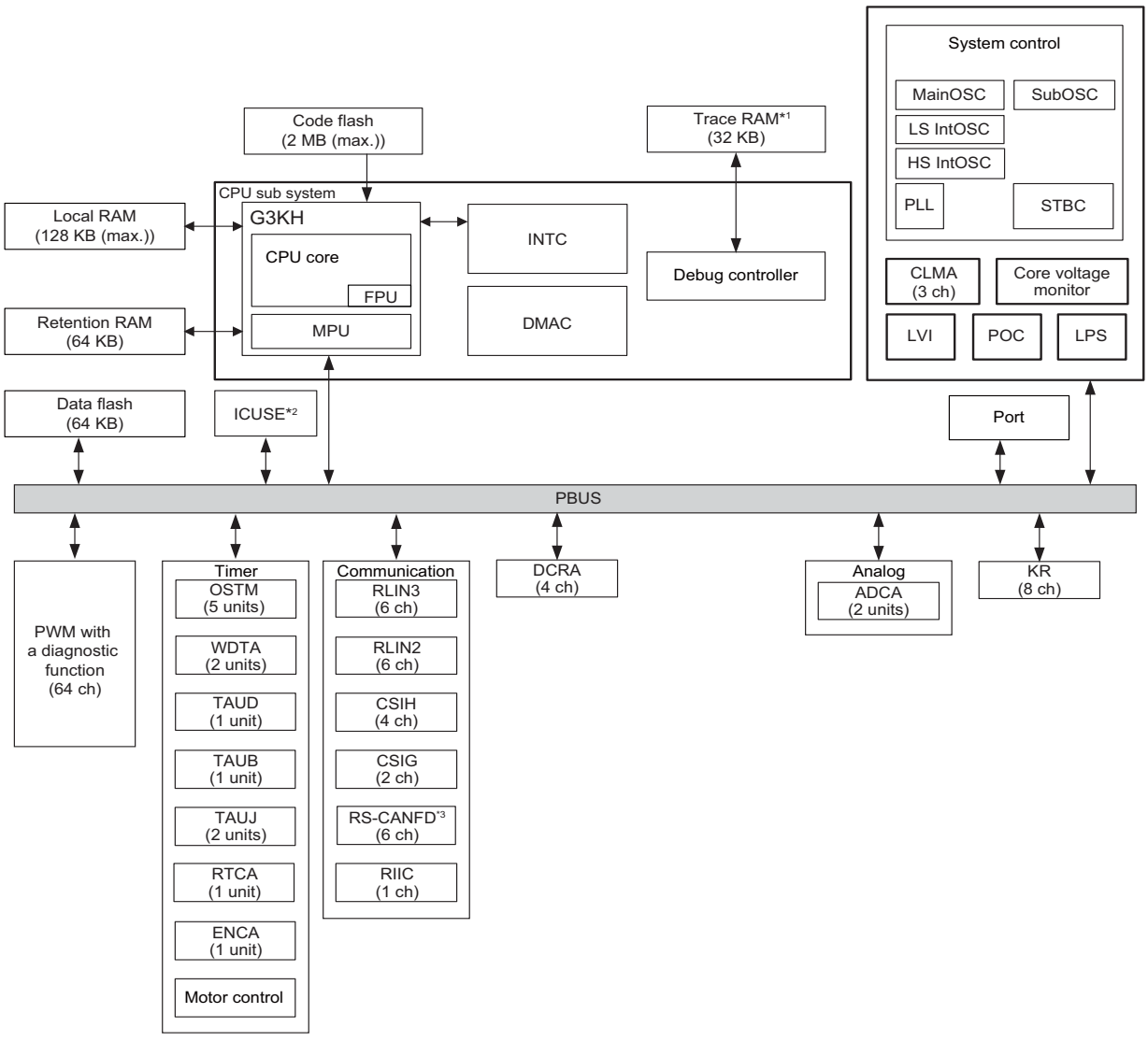
| Pin Count | CPU frequency | Memory | | | | CAN Interface (RSCAN) | CAN FD Interface (RS-CANFD) | Part Name | | Line Name |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Code Flash | Local RAM | Data Flash | Retention RAM (RRAM) | | | Operating Temperature (Ta) | | |
| | | | | | | | | –40°C to +105°C | –40°C to +125°C | |
| 176 pins | 120 MHz max. | 1024 KB | 64 KB | 64 KB | 64 KB | 7 channels | — | R7F7015773AFP<br>R7F7015773AFP-C | R7F7015774AFP<br>R7F7015774AFP-C | ADVANCED |
| | | 1536 KB | 96 KB | | | | | R7F7015663AFP<br>R7F7015663AFP-C | R7F7015664AFP<br>R7F7015664AFP-C | |
| | | 2048 KB | 128 KB | | | | | R7F7015673AFP<br>R7F7015673AFP-C | R7F7015674AFP<br>R7F7015674AFP-C | |
| 100 pins | 120 MHz max. | 768 KB | 32 KB | 64 KB | 64 KB | — | 6 channels | R7F7016203AFP<br>R7F7016203AFP-C | R7F7016204AFP<br>R7F7016204AFP-C | PREMIUM |
| | | 1024 KB | 64 KB | | | | | R7F7016213AFP<br>R7F7016213AFP-C | R7F7016214AFP<br>R7F7016214AFP-C | |
| | | 1536 KB | 96 KB | | | | | R7F7015803AFP<br>R7F7015803AFP-C | R7F7015804AFP<br>R7F7015804AFP-C | |
| | | 2048 KB | 128 KB | | | | | R7F7015813AFP<br>R7F7015813AFP-C | R7F7015814AFP<br>R7F7015814AFP-C | |
| 144pins | 120 MHz max. | 768 KB | 32 KB | 64 KB | 64 KB | — | 6 channels | R7F7016223AFP<br>R7F7016223AFP-C | R7F7016224AFP<br>R7F7016224AFP-C | |
| | | 1024 KB | 64 KB | | | | | R7F7016233AFP<br>R7F7016233AFP-C | R7F7016234AFP<br>R7F7016234AFP-C | |
| | | 1536 KB | 96 KB | | | | | R7F7015823AFP<br>R7F7015823AFP-C | R7F7015824AFP<br>R7F7015824AFP-C | |
| | | 2048 KB | 128 KB | | | | | R7F7015833AFP<br>R7F7015833AFP-C | R7F7015834AFP<br>R7F7015834AFP-C | |
| 176pins | 120 MHz max. | 1024 KB | 64 KB | 64 KB | 64 KB | 1 channel | 6 channels | R7F7015973AFP<br>R7F7015973AFP-C | R7F7015974AFP<br>R7F7015974AFP-C | |
| | | 1536 KB | 96 KB | | | | | R7F7015863AFP<br>R7F7015863AFP-C | R7F7015864AFP<br>R7F7015864AFP-C | |
| | | 2048 KB | 128 KB | | | | | R7F7015873AFP<br>R7F7015873AFP-C | R7F7015874AFP<br>R7F7015874AFP-C | |

## 1.4 RH850/F1K Product Block Diagrams



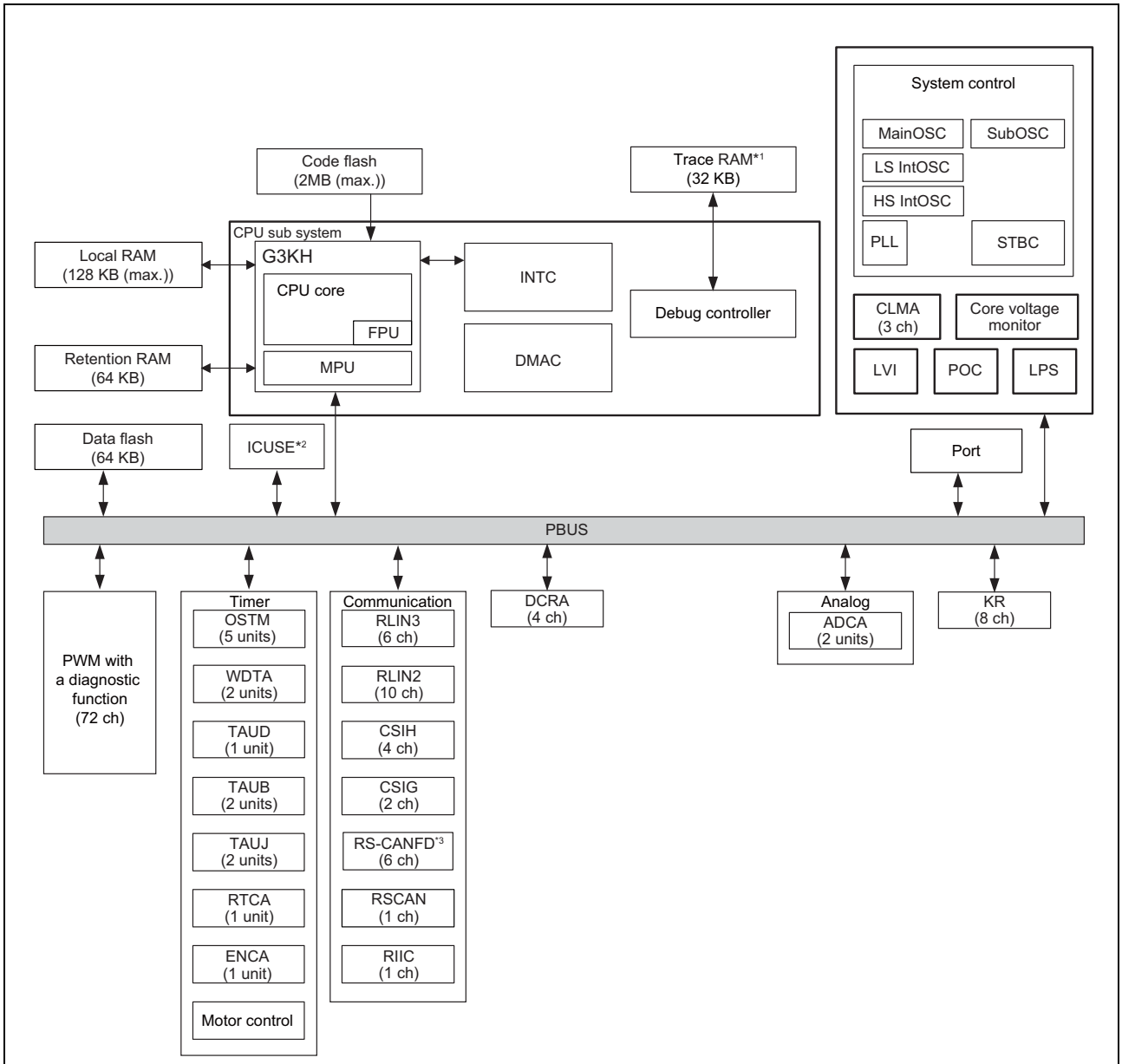

Note 1. The trace RAM is only supported by products with 2 MB code flash memory.
Note 2. ADVANCED: 10 keys, PREMIUM: 20 keys.
Note 3. PREMIUM only. RSCAN is used in ADVANCED.

**Figure 1.1 Internal Block Diagram (RH850/F1K 100 pin)**




Note 1. The trace RAM is only supported by products with 2 MB code flash memory.
Note 2. ECO/ADVANCED: 10 keys, PREMIUM: 20 keys.
Note 3. PREMIUM only. RSCAN is used in ECO and ADVANCED.

**Figure 1.2 Internal Block Diagram (RH850/F1K 144 pin)**




Note 1. The trace RAM is only supported by products with 2 MB code flash memory.
Note 2. ECO/ADVANCED: 10 keys, PREMIUM: 20 keys.
Note 3. PREMIUM only. RSCAN is used in ECO and ADVANCED.

**Figure 1.3 Internal Block Diagram (RH850/F1K 176 pin)**

# Section 2 Pin Function

This section describes the pin and port functions.

**Section 2.1** to **Section 2.5** describe the pin connections and respective pins.

**Section 2.6** to **Section 2.13** describe the general port functions.

## 2.1 Pin Connection Diagram

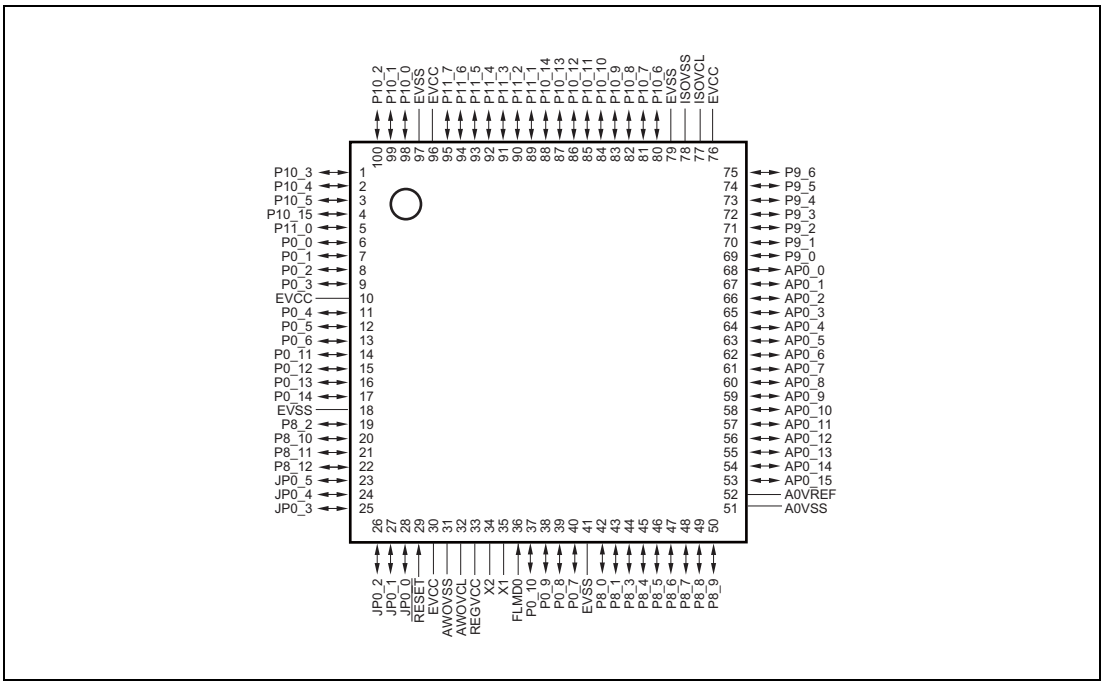



**Figure 2.1 Pin Connection Diagram (100 Pin LQFP)**

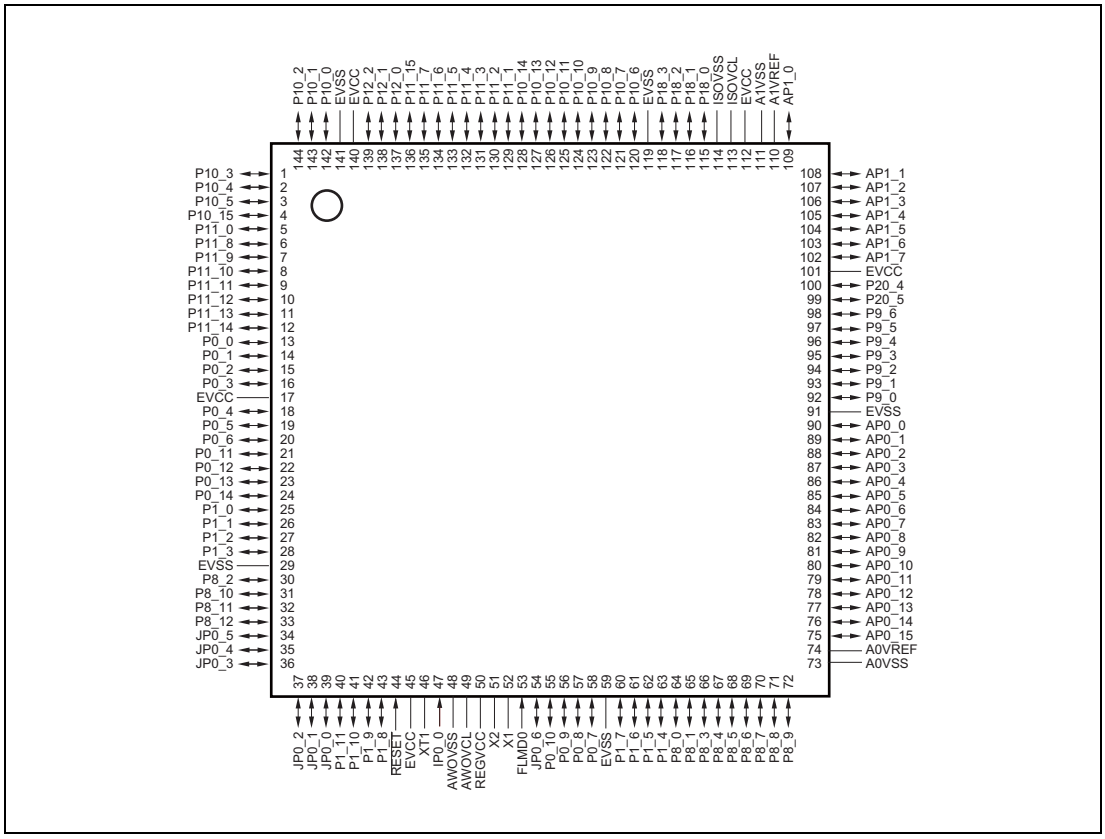



**Figure 2.2 Pin Connection Diagram (144 Pin LQFP)**

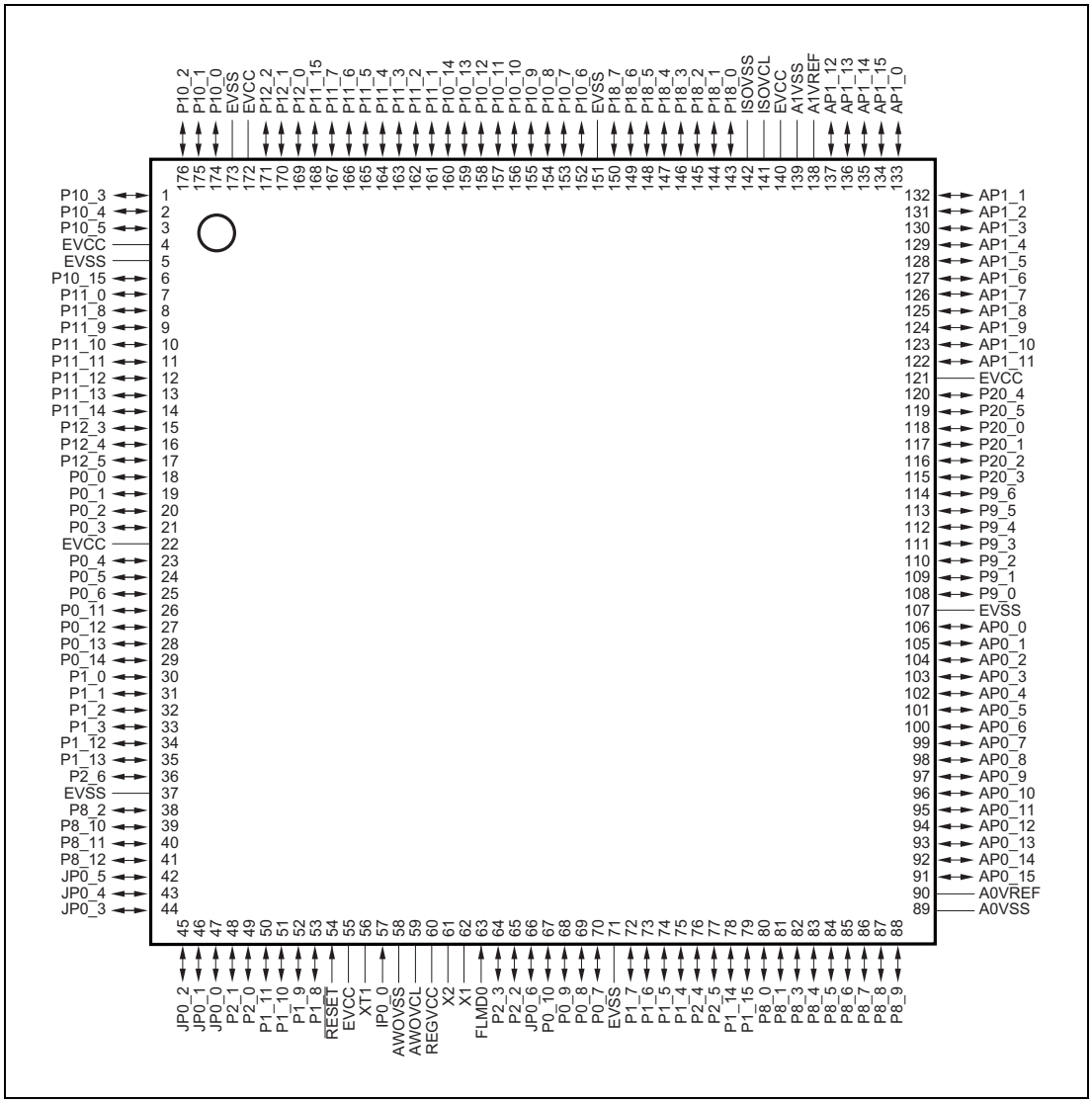



**Figure 2.3 Pin Connection Diagram (176 Pin LQFP)**

**Table 2.1 Pin Assignment 100 Pin LQFP (1/3)**

| Pin No. | Pin Name |
|---|---|
| 1 | P10_3 / TAUD0I7 / TAUD0O7 / RIIC0SCL / KR0I1 / PWGA3O / ADCA0TRG1 / TAPA0VN / $\overline{\text{CSIH1SSI}}$ |
| 2 | P10_4 / TAUD0I9 / TAUD0O9 / RLIN21RX / KR0I2 / ADCA0SEL0 / ADCA0TRG2 / TAPA0WP / $\overline{\text{CSIG0SSI}}$ |
| 3 | P10_5 / TAUD0I11 / TAUD0O11 / RLIN21TX / KR0I3 / ADCA0SEL1 / TAPA0WN / CSIG0RYI / CSIG0RYO |
| 4 | P10_15 / CSIH3RYI / CSIH3RYO / PWGA24O / RLIN22RX / TAUB0I9 / TAUB0O9 |
| 5 | P11_0 / CSIH2RYI / CSIH2RYO / PWGA25O / RLIN22TX / TAUB0I11 / TAUB0O11 |
| 6 | P0_0 / TAUD0I2 / TAUD0O2 / RLIN20RX / CAN0TX / PWGA10O / $\overline{\text{CSIH0SSI}}$ / DPO |
| 7 | P0_1 / TAUD0I4 / TAUD0O4 / CAN0RX / INTP0 / RLIN20TX / PWGA11O / CSIH0SI / APO |
| 8 | P0_2 / TAUD0I6 / TAUD0O6 / CAN1RX / INTP1 / RLIN30TX / PWGA12O / CSIH0SC / DPO |
| 9 | P0_3 / TAUD0I8 / TAUD0O8 / RLIN30RX / INTP10 / CAN1TX / DPIN1 / PWGA13O / CSIH0SO |
| 10 | EVCC |
| 11 | P0_4 / RLIN31RX / INTP11 / CAN2TX / PWGA10O / CSIH1SI / SELDP0 /DPIN8 |
| 12 | P0_5 / CAN2RX / INTP2 / RLIN31TX / DPIN9 / SELDP1 / CSIH1SO |
| 13 | P0_6 / INTP2 / DPIN10 / SELDP2 / CSIH1SC / PWGA35O |
| 14 | P0_11 / RIIC0SDA / DPIN12 / CSIH1CSS2 / TAUB0I8 / TAUB0O8 / PWGA34O |
| 15 | P0_12 / RIIC0SCL / DPIN13 / PWGA45O / TAUB0I10 / TAUB0O10 / CSIG0SI |
| 16 | P0_13 / RLIN32RX / INTP12 / PWGA46O / TAUB0I12 / TAUB0O12 / CSIG0SO / CAN5RX / INTP5 |
| 17 | P0_14 / RLIN32TX / PWGA47O / TAUB0I14 / TAUB0O14 / CSIG0SC / CAN5TX |
| 18 | EVSS |
| 19 | P8_2 / TAUJ0I0 / TAUJ0O0 / DPIN2 / CSIH0CSS0 / INTP6 / PWGA22O / ADCA0I4S |
| 20 | P8_10 / CSIH3CSS3 / DPIN14 / PWGA42O / ADCA0I17S |
| 21 | P8_11 / TAUJ1I2 / TAUJ1O2 / DPIN15 / PWGA43O / CSIH1CSS4 / ADCA0I18S |
| 22 | P8_12 / TAUJ1I3 / TAUJ1O3 / DPIN16 / PWGA44O / CSIH1CSS5 / ADCA0I19S |
| 23 | JP0_5 / NMI / TAUJ0I3 / TAUJ0O3 / $\overline{\text{DCURDY}}$ / LPDCLKOUT |
| 24 | JP0_4 / $\overline{\text{DCUTRST}}$ |
| 25 | JP0_3 / INTP3 / CSCXFOUT / TAUJ0I2 / TAUJ0O2 / DCUTMS |
| 26 | JP0_2 / INTP2 / TAUJ0I1 / TAUJ0O1 / FPCK / DCUTCK / LPDCLK |
| 27 | JP0_1 / INTP1 /TAUJ0I0 / TAUJ0O0 / FPDT / DCUTDO / LPDO |
| 28 | JP0_0 / INTP0 / FPDR / FPDT / DCUTDI / LPDI / LPDIO |
| 29 | $\overline{\text{RESET}}$ |
| 30 | EVCC |
| 31 | AWOVSS |
| 32 | AWOVCL |
| 33 | REGVCC |
| 34 | X2 |
| 35 | X1 |
| 36 | FLMD0 |
| 37 | P0_10 / INTP3 / CSIH1CSS1 / DPIN11 / RLIN22TX / TAUB0I6 / TAUB0O6 / CAN4TX |
| 38 | P0_9 / INTP12 / CSIH1CSS0 / DPIN7 / RLIN22RX / TAUB0I4 / TAUB0O4 / CAN4RX / INTP4 |
| 39 | P0_8/ RLIN21TX / DPIN6 / CSIH0CSS6 / $\overline{\text{CSIH1SSI}}$ / TAUB0I2 / TAUB0O2 / CAN3TX |
| 40 | P0_7 / RLIN21RX / DPIN5 / CSCXFOUT / CSIH1RYI / CSIH1RYO / TAUB0I0 / TAUB0O0 / CAN3RX / INTP3 |
| 41 | EVSS |

**Table 2.1 Pin Assignment 100 Pin LQFP (2/3)**

| Pin No. | Pin Name |
|---|---|
| 42 | P8_0 / TAUJ0I0 / TAUJ0O0 / DPIN2 / PWGA14O / INTP4 / CSIH0CSS0 / ADCA0I0S |
| 43 | P8_1 / TAPA0ESO / TAUJ0O1 / DPIN0 / PWGA15O / INTP5 / CSIH1CSS3 / ADCA0I1S |
| 44 | P8_3 / TAUJ0I1 / TAUJ0O1 / DPIN3 / CSIH0CSS1 / INTP7 / PWGA23O / ADCA0I5S |
| 45 | P8_4 / TAUJ0I2 / TAUJ0O2 / DPIN4 / CSIH0CSS2 / INTP8 / PWGA36O / ADCA0I6S |
| 46 | P8_5 / TAUJ0I3 / TAUJ0O3 / NMI / CSIH0CSS3 / PWGA37O / ADCA0I7S |
| 47 | P8_6 / NMI / CSIH0CSS4 / PWGA38O / ADCA0I8S / $\overline{\text{RESETOUT}}$ |
| 48 | P8_7 / CSIH3CSS0 / PWGA39O / ADCA0SEL0 / ADCA0I14S |
| 49 | P8_8 / CSIH3CSS1 / PWGA40O / ADCA0SEL1 / ADCA0I15S |
| 50 | P8_9 / CSIH3CSS2 / PWGA41O / ADCA0SEL2 / ADCA0I16S |
| 51 | A0VSS |
| 52 | A0VREF |
| 53 | AP0_15 / ADCA0I15 |
| 54 | AP0_14 / ADCA0I14 |
| 55 | AP0_13 / ADCA0I13 |
| 56 | AP0_12 / ADCA0I12 |
| 57 | AP0_11 / ADCA0I11 |
| 58 | AP0_10 / ADCA0I10 |
| 59 | AP0_9 / ADCA0I9 |
| 60 | AP0_8 / ADCA0I8 |
| 61 | AP0_7 / ADCA0I7 |
| 62 | AP0_6 / ADCA0I6 |
| 63 | AP0_5 / ADCA0I5 |
| 64 | AP0_4 / ADCA0I4 |
| 65 | AP0_3 / ADCA0I3 |
| 66 | AP0_2 / ADCA0I2 |
| 67 | AP0_1 / ADCA0I1 |
| 68 | AP0_0 / ADCA0I0 |
| 69 | P9_0 / NMI / PWGA8O / TAUD0I0 / TAUD0O0 / ADCA0TRG0 / CSIH2CSS0 / KR0I4 / ADCA0I2S |
| 70 | P9_1 / INTP11 / PWGA9O / TAUD0I2 / TAUD0O2 / KR0I5 / CSIH2CSS1 / ADCA0I3S |
| 71 | P9_2 / KR0I6 / PWGA20O / TAPA0ESO / CSIH2CSS2 / ADCA0I9S |
| 72 | P9_3 / KR0I7 / PWGA21O / CSIH2CSS3 / TAUJ1I1 / TAUJ1O1 / ADCA0I10S |
| 73 | P9_4 / CSIH0CSS5 / PWGA33O / TAUJ1I0 / TAUJ1O0 / ADCA0I11S |
| 74 | P9_5 / CSIH0CSS6 / PWGA34O / TAUJ1I1 / TAUJ1O1 / ADCA0I12S |
| 75 | P9_6 / CSIH0CSS7 / PWGA35O / ADCA0I13S |
| 76 | EVCC |
| 77 | ISOVCL |
| 78 | ISOVSS |
| 79 | EVSS |
| 80 | P10_6 / TAUD0I13 / TAUD0O13 / CSIG0SO / ENCA0TIN0 / ADCA0SEL2 / CAN1RX /INTP1 / MODE2 |
| 81 | P10_7 / TAUD0I15 / TAUD0O15 / CSIG0SC / ENCA0TIN1 / PWGA4O / CAN1TX |
| 82 | P10_8 / TAUD0I10 / TAUD0O10 / CSIG0SI / ENCA0EC / PWGA5O / FLMD1 |

**Table 2.1 Pin Assignment 100 Pin LQFP (3/3)**

| Pin No. | Pin Name |
|---|---|
| 83 | P10_9 / TAUD0I12 / TAUD0O12 / RLIN30RX /INTP10 / ENCA0E0 / PWGA6O / CSIH0RYI / CSIH0RYO |
| 84 | P10_10 / TAUD0I14 / TAUD0O14 / RLIN30TX / ENCA0E1 / PWGA7O / CSIH0CSS1 |
| 85 | P10_11 / PWGA16O / RLIN31RX / INTP11 / CSIH1CSS0 / TAUB0I1 / TAUB0O1 |
| 86 | P10_12 / PWGA17O / RLIN31TX / CSIH1CSS1 / TAUB0I3 / TAUB0O3 |
| 87 | P10_13 / $\overline{\text{CSIH0SSI}}$ / PWGA18O / RLIN32RX / INTP12 / TAUB0I5 / TAUB0O5 |
| 88 | P10_14 / PWGA19O / RLIN32TX / $\overline{\text{CSIH3SSI}}$ / TAUB0I7 / TAUB0O7 |
| 89 | P11_1 / $\overline{\text{CSIH2SSI}}$ / RLIN20RX / CSIH0CSS7 / PWGA26O / TAUB0I13 / TAUB0O13 |
| 90 | P11_2 / CSIH2SO / RLIN32RX / INTP12 / RLIN20TX / PWGA27O / TAUB0I15 / TAUB0O15 |
| 91 | P11_3 / CSIH2SC / CAN3RX / INTP3 / PWGA28O / RLIN32TX |
| 92 | P11_4 / CSIH2SI / CAN3TX / PWGA29O |
| 93 | P11_5 / CAN5RX / INTP5 / RLIN33TX / PWGA30O / CSIH3SI |
| 94 | P11_6 / RLIN33RX / INTP13 / CAN5TX / PWGA31O / CSIH3SO |
| 95 | P11_7 / INTP5 / PWGA32O / CSIH3SC |
| 96 | EVCC |
| 97 | EVSS |
| 98 | P10_0 / TAUD0I1 / TAUD0O1 / CAN0RX /INTP0 / CSCXFOUT / PWGA0O / TAPA0UP / CSIH1SI |
| 99 | P10_1 / TAUD0I3 / TAUD0O3 / CAN0TX / PWGA1O / TAPA0UN / CSIH1SC / MODE0 |
| 100 | P10_2 / TAUD0I5 / TAUD0O5 / RIIC0SDA / KR0I0 / PWGA2O / ADCA0TRG0 / TAPA0VP / CSIH1SO / MODE1 |

**Table 2.2 Pin Assignment 144 Pin LQFP (1/4)**

| Pin No. | Pin Name |
|---|---|
| 1 | P10_3 / TAUD0I7 / TAUD0O7 / RIIC0SCL / KR0I1 / PWGA3O / ADCA0TRG1 / TAPA0VN / $\overline{\text{CSIH1SSI}}$ |
| 2 | P10_4 / TAUD0I9 / TAUD0O9 / RLIN21RX / KR0I2 / ADCA0SEL0 / ADCA0TRG2 / TAPA0WP / $\overline{\text{CSIG0SSI}}$ |
| 3 | P10_5 / TAUD0I11 / TAUD0O11 / RLIN21TX / KR0I3 / ADCA0SEL1 / TAPA0WN / CSIG0RYI / CSIG0RYO |
| 4 | P10_15 / CSIH3RYI / CSIH3RYO / PWGA24O / RLIN22RX / TAUB0I9 / TAUB0O9 |
| 5 | P11_0 / CSIH2RYI / CSIH2RYO / ADCA1TRG2 / PWGA25O / RLIN22TX / TAUB0I11 / TAUB0O11 |
| 6 | P11_8 / $\overline{\text{CSIG1SSI}}$ / RLIN35TX / PWGA48O |
| 7 | P11_9 / CSIG1SO / RLIN35RX / INTP15 / PWGA49O |
| 8 | P11_10 / CSIG1SC / PWGA50O |
| 9 | P11_11 / CSIG1SI / RLIN25TX / PWGA51O |
| 10 | P11_12 / RLIN25RX / PWGA52O |
| 11 | P11_13 / RLIN24RX / PWGA53O |
| 12 | P11_14 / RLIN24TX / PWGA54O |
| 13 | P0_0 / TAUD0I2 / TAUD0O2 / RLIN20RX / CAN0TX / PWGA10O / $\overline{\text{CSIH0SSI}}$ / DPO |
| 14 | P0_1 / TAUD0I4 / TAUD0O4 / CAN0RX / INTP0 / RLIN20TX / PWGA11O / CSIH0SI / APO |
| 15 | P0_2 / TAUD0I6 / TAUD0O6 / CAN1RX / INTP1 / RLIN30TX / PWGA12O / CSIH0SC / DPO |
| 16 | P0_3 / TAUD0I8 / TAUD0O8 / RLIN30RX / INTP10 / CAN1TX / DPIN1 / PWGA13O / CSIH0SO |
| 17 | EVCC |
| 18 | P0_4 / RLIN31RX / INTP11 / CAN2TX / PWGA10O / CSIH1SI / SELDP0 /DPIN8 |
| 19 | P0_5 / CAN2RX / INTP2 / RLIN31TX / DPIN9 / SELDP1 / CSIH1SO |
| 20 | P0_6 / INTP2 / DPIN10 / SELDP2 / CSIH1SC / PWGA35O |
| 21 | P0_11 / RIIC0SDA / DPIN12 / CSIH1CSS2 / TAUB0I8 / TAUB0O8 / PWGA34O |
| 22 | P0_12 / RIIC0SCL / DPIN13 / PWGA45O / TAUB0I10 / TAUB0O10 / CSIG0SI |
| 23 | P0_13 / RLIN32RX / INTP12 / PWGA46O / TAUB0I12 / TAUB0O12 / CSIG0SO / CAN5RX / INTP5 |
| 24 | P0_14 / RLIN32TX / PWGA47O / TAUB0I14 / TAUB0O14 / CSIG0SC / CAN5TX |
| 25 | P1_0 / RLIN33RX / INTP13 |
| 26 | P1_1 / RLIN33TX |
| 27 | P1_2 / CAN3RX / INTP3 |
| 28 | P1_3 / CAN3TX / DPIN23 |
| 29 | EVSS |
| 30 | P8_2 / TAUJ0I0 / TAUJ0O0 / DPIN2 / CSIH0CSS0 / INTP6 / PWGA22O / ADCA0I4S |
| 31 | P8_10 / CSIH3CSS3 / DPIN14 / PWGA42O / ADCA0I17S |
| 32 | P8_11 / TAUJ1I2 / TAUJ1O2 / DPIN15 / PWGA43O / CSIH1CSS4 / ADCA0I18S |
| 33 | P8_12 / TAUJ1I3 / TAUJ1O3 / DPIN16 / PWGA44O / CSIH1CSS5 / ADCA0I19S |
| 34 | JP0_5 / NMI / RTCA0OUT / TAUJ0I3 / TAUJ0O3 / $\overline{\text{DCURDY}}$ / LPDCLKOUT |
| 35 | JP0_4 / $\overline{\text{DCUTRST}}$ |
| 36 | JP0_3 / INTP3 / CSCXFOUT / TAUJ0I2 / TAUJ0O2 / DCUTMS |
| 37 | JP0_2 / INTP2 / TAUJ0I1 / TAUJ0O1 / FPCK / DCUTCK / LPDCLK |
| 38 | JP0_1 / INTP1 /TAUJ0I0 / TAUJ0O0 / FPDT / DCUTDO / LPDO |
| 39 | JP0_0 / INTP0 / FPDR / FPDT / DCUTDI / LPDI / LPDIO |
| 40 | P1_11 / ADCA1TRG2 / RLIN24TX / DPIN22 |
| 41 | P1_10 / RLIN24RX / DPIN21 |

**Table 2.2 Pin Assignment 144 Pin LQFP (2/4)**

| Pin No. | Pin Name |
|---|---|
| 42 | P1_9 / RLIN34TX / DPIN20 |
| 43 | P1_8 / RLIN34RX / INTP14 |
| 44 | $\overline{\text{RESET}}$ |
| 45 | EVCC |
| 46 | XT1 |
| 47 | IP0_0 / XT2 |
| 48 | AWOVSS |
| 49 | AWOVCL |
| 50 | REGVCC |
| 51 | X2 |
| 52 | X1 |
| 53 | FLMD0 |
| 54 | JP0_6 / $\overline{\text{EVTO}}$ |
| 55 | P0_10 / INTP3 / CSIH1CSS1 / DPIN11 / RLIN22TX / TAUB0I6 / TAUB0O6 / CAN4TX |
| 56 | P0_9 / INTP12 / CSIH1CSS0 / DPIN7 / RLIN22RX / TAUB0I4 / TAUB0O4 / CAN4RX / INTP4 |
| 57 | P0_8/ RLIN21TX / DPIN6 / CSIH0CSS6 / $\overline{\text{CSIH1SSI}}$ / TAUB0I2 / TAUB0O2 / CAN3TX |
| 58 | P0_7 / RLIN21RX / DPIN5 / CSCXFOUT / CSIH1RYI / CSIH1RYO / TAUB0I0 / TAUB0O0 / CAN3RX / INTP3 |
| 59 | EVSS |
| 60 | P1_7 / ADCA1TRG1 / RLIN25TX / DPIN19 |
| 61 | P1_6 / RLIN25RX / DPIN18 |
| 62 | P1_5 / ADCA1TRG0 / RLIN35TX / DPIN17 |
| 63 | P1_4 / RLIN35RX / INTP15 |
| 64 | P8_0 / TAUJ0I0 / TAUJ0O0 / DPIN2 / PWGA14O / INTP4 / CSIH0CSS0 / ADCA0I0S |
| 65 | P8_1 / TAPA0ESO / TAUJ0O1 / DPIN0 / PWGA15O / INTP5 / CSIH1CSS3 / ADCA0I1S |
| 66 | P8_3 / TAUJ0I1 / TAUJ0O1 / DPIN3 / CSIH0CSS1 / INTP7 / PWGA23O / ADCA0I5S |
| 67 | P8_4 / TAUJ0I2 / TAUJ0O2 / DPIN4 / CSIH0CSS2 / INTP8 / PWGA36O / ADCA0I6S |
| 68 | P8_5 / TAUJ0I3 / TAUJ0O3 / NMI / CSIH0CSS3 / INTP9 / PWGA37O / ADCA0I7S |
| 69 | P8_6 / NMI / CSIH0CSS4 / PWGA38O / RTCA0OUT / ADCA0I8S / $\overline{\text{RESETOUT}}$ |
| 70 | P8_7 / CSIH3CSS0 / PWGA39O / ADCA0SEL0 / RTCA0OUT / ADCA0I14S |
| 71 | P8_8 / CSIH3CSS1 / PWGA40O / ADCA0SEL1 / ADCA0I15S |
| 72 | P8_9 / CSIH3CSS2 / PWGA41O / ADCA0SEL2 / ADCA0I16S |
| 73 | A0VSS |
| 74 | A0VREF |
| 75 | AP0_15 / ADCA0I15 |
| 76 | AP0_14 / ADCA0I14 |
| 77 | AP0_13 / ADCA0I13 |
| 78 | AP0_12 / ADCA0I12 |
| 79 | AP0_11 / ADCA0I11 |
| 80 | AP0_10 / ADCA0I10 |
| 81 | AP0_9 / ADCA0I9 |
| 82 | AP0_8 / ADCA0I8 |

Table 2.2 Pin Assignment 144 Pin LQFP (3/4)

| Pin No. | Pin Name |
|---|---|
| 83 | AP0_7 / ADCA0I7 |
| 84 | AP0_6 / ADCA0I6 |
| 85 | AP0_5 / ADCA0I5 |
| 86 | AP0_4 / ADCA0I4 |
| 87 | AP0_3 / ADCA0I3 |
| 88 | AP0_2 / ADCA0I2 |
| 89 | AP0_1 / ADCA0I1 |
| 90 | AP0_0 / ADCA0I0 |
| 91 | EVSS |
| 92 | P9_0 / NMI / PWGA8O / TAUD0I0 / TAUD0O0 / ADCA0TRG0 / CSIH2CSS0 / KR0I4 / ADCA0I2S |
| 93 | P9_1 / INTP11 / PWGA9O / TAUD0I2 / TAUD0O2 / KR0I5 / CSIH2CSS1 / ADCA0I3S |
| 94 | P9_2 / KR0I6 / PWGA20O / TAPA0ESO / CSIH2CSS2 / ADCA0I9S |
| 95 | P9_3 / KR0I7 / PWGA21O / CSIH2CSS3 / TAUJ1I1 / TAUJ1O1 / ADCA0I10S |
| 96 | P9_4 / CSIH0CSS5 / PWGA33O / TAUJ1I0 / TAUJ1O0 / ADCA0I11S |
| 97 | P9_5 / CSIH0CSS6 / PWGA34O / TAUJ1I1 / TAUJ1O1 / ADCA0I12S |
| 98 | P9_6 / CSIH0CSS7 / PWGA35O / ADCA0I13S |
| 99 | P20_5 / RLIN23TX / PWGA60O |
| 100 | P20_4 / RLIN23RX / PWGA59O |
| 101 | EVCC |
| 102 | AP1_7 / ADCA1I7 |
| 103 | AP1_6 / ADCA1I6 |
| 104 | AP1_5 / ADCA1I5 |
| 105 | AP1_4 / ADCA1I4 |
| 106 | AP1_3 / ADCA1I3 |
| 107 | AP1_2 / ADCA1I2 |
| 108 | AP1_1 / ADCA1I1 |
| 109 | AP1_0 / ADCA1I0 |
| 110 | A1VREF |
| 111 | A1VSS |
| 112 | EVCC |
| 113 | ISOVCL |
| 114 | ISOVSS |
| 115 | P18_0 / CSIG1RYI / CSIG1RYO / PWGA61O / ADCA1I0S |
| 116 | P18_1 / PWGA62O / ADCA1I1S |
| 117 | P18_2 / PWGA63O / ADCA1I2S |
| 118 | P18_3 / ADCA1I3S |
| 119 | EVSS |
| 120 | P10_6 / TAUD0I13 / TAUD0O13 / CSIG0SO / ENCA0TIN0 / ADCA0SEL2 / CAN1RX /INTP1 / MODE2 |
| 121 | P10_7 / TAUD0I15 / TAUD0O15 / CSIG0SC / ENCA0TIN1 / PWGA4O / CAN1TX |
| 122 | P10_8 / TAUD0I10 / TAUD0O10 / CSIG0SI / ENCA0EC / PWGA5O / FLMD1 |
| 123 | P10_9 / TAUD0I12 / TAUD0O12 / RLIN30RX / INTP10 / ENCA0E0 / PWGA6O / CSIH0RYI / CSIH0RYO |

**Table 2.2 Pin Assignment 144 Pin LQFP (4/4)**

| Pin No. | Pin Name |
|---|---|
| 124 | P10_10 / TAUD0I14 / TAUD0O14 / RLIN30TX / ENCA0E1 / PWGA7O / CSIH0CSS1 |
| 125 | P10_11 / PWGA16O / RLIN31RX / INTP11 / CSIH1CSS0 / TAUB0I1 / TAUB0O1 |
| 126 | P10_12 / PWGA17O / RLIN31TX / CSIH1CSS1 / TAUB0I3 / TAUB0O3 |
| 127 | P10_13 / $\overline{\text{CSIH0SSI}}$ / PWGA18O / RLIN32RX / INTP12 / TAUB0I5 / TAUB0O5 |
| 128 | P10_14 / ADCA1TRG0 / PWGA19O / RLIN32TX / $\overline{\text{CSIH3SSI}}$ / TAUB0I7 / TAUB0O7 |
| 129 | P11_1 / $\overline{\text{CSIH2SSI}}$ / RLIN20RX / CSIH0CSS7 / PWGA26O / TAUB0I13 / TAUB0O13 |
| 130 | P11_2 / CSIH2SO / RLIN32RX / INTP12 / RLIN20TX / PWGA27O / TAUB0I15 / TAUB0O15 |
| 131 | P11_3 / CSIH2SC / CAN3RX / INTP3 / PWGA28O / RLIN32TX |
| 132 | P11_4 / CSIH2SI / CAN3TX / PWGA29O |
| 133 | P11_5 / CAN5RX / INTP5 / RLIN33TX / PWGA30O / CSIH3SI |
| 134 | P11_6 / RLIN33RX / INTP13 / CAN5TX / ADCA1TRG1 / PWGA31O / CSIH3SO |
| 135 | P11_7 / INTP5 / PWGA32O / CSIH3SC |
| 136 | P11_15 / CAN2RX / INTP2 / CSIH2CSS4 / PWGA55O |
| 137 | P12_0 / CAN2TX / PWGA56O |
| 138 | P12_1 / RLIN34RX / INTP14 / CSIH2CSS5 / PWGA57O |
| 139 | P12_2 / RLIN34TX / PWGA58O |
| 140 | EVCC |
| 141 | EVSS |
| 142 | P10_0 / TAUD0I1 / TAUD0O1 / CAN0RX /INTP0 / CSCXFOUT / PWGA0O / TAPA0UP / CSIH1SI |
| 143 | P10_1 / TAUD0I3 / TAUD0O3 / CAN0TX / PWGA1O / TAPA0UN / CSIH1SC / MODE0 |
| 144 | P10_2 / TAUD0I5 / TAUD0O5 / RIIC0SDA / KR0I0 / PWGA2O / ADCA0TRG0 / TAPA0VP / CSIH1SO / MODE1 |

**Table 2.3 Pin Assignment 176 Pin LQFP (1/5)**

| Pin No. | Pin Name |
|---|---|
| 1 | P10_3 / TAUD0I7 / TAUD0O7 / RIIC0SCL / KR0I1 / PWGA3O / ADCA0TRG1 / TAPA0VN / $\overline{\text{CSIH1SSI}}$ |
| 2 | P10_4 / TAUD0I9 / TAUD0O9 / RLIN21RX / KR0I2 / ADCA0SEL0 / ADCA0TRG2 / TAPA0WP / $\overline{\text{CSIG0SSI}}$ |
| 3 | P10_5 / TAUD0I11 / TAUD0O11 / RLIN21TX / KR0I3 / ADCA0SEL1 / TAPA0WN / CSIG0RYI / CSIG0RYO |
| 4 | EVCC |
| 5 | EVSS |
| 6 | P10_15 / CSIH3RYI / CSIH3RYO / PWGA24O / RLIN22RX / TAUB0I9 / TAUB0O9 |
| 7 | P11_0 / CSIH2RYI / CSIH2RYO / ADCA1TRG2 / PWGA25O / RLIN22TX / TAUB0I11 / TAUB0O11 |
| 8 | P11_8 / $\overline{\text{CSIG1SSI}}$ / RLIN35TX / PWGA48O / TAUB1I11 / TAUB1O11 |
| 9 | P11_9 / CSIG1SO / RLIN35RX / INTP15 / PWGA49O / TAUB1I13 / TAUB1O13 |
| 10 | P11_10 / CSIG1SC / PWGA50O / TAUB1I15 / TAUB1O15 |
| 11 | P11_11 / CSIG1SI / RLIN25TX / PWGA51O / TAUB1I0 / TAUB1O0 |
| 12 | P11_12 / RLIN25RX / PWGA52O / TAUB1I2 / TAUB1O2 |
| 13 | P11_13 / RLIN24RX / PWGA53O / TAUB1I4 / TAUB1O4 |
| 14 | P11_14 / RLIN24TX / PWGA54O / TAUB1I6 / TAUB1O6 |
| 15 | P12_3 / RLIN27RX / PWGA68O |
| 16 | P12_4 / RLIN27TX / PWGA69O |
| 17 | P12_5 / PWGA70O |
| 18 | P0_0 / TAUD0I2 / TAUD0O2 / RLIN20RX / CAN0TX / PWGA10O / $\overline{\text{CSIH0SSI}}$ / DPO |
| 19 | P0_1 / TAUD0I4 / TAUD0O4 / CAN0RX / INTP0 / RLIN20TX / PWGA11O / CSIH0SI / APO |
| 20 | P0_2 / TAUD0I6 / TAUD0O6 / CAN1RX / INTP1 / RLIN30TX / PWGA12O / CSIH0SC / DPO |
| 21 | P0_3 / TAUD0I8 / TAUD0O8 / RLIN30RX / INTP10 / CAN1TX / DPIN1 / PWGA13O / CSIH0SO |
| 22 | EVCC |
| 23 | P0_4 / RLIN31RX / INTP11 / CAN2TX / PWGA10O / CSIH1SI / SELDP0 /DPIN8 |
| 24 | P0_5 / CAN2RX / INTP2 / RLIN31TX / DPIN9 / SELDP1 / CSIH1SO |
| 25 | P0_6 / INTP2 / DPIN10 / SELDP2 / CSIH1SC / PWGA35O |
| 26 | P0_11 / RIIC0SDA / DPIN12 / CSIH1CSS2 / TAUB0I8 / TAUB0O8 / RLIN26RX / PWGA34O |
| 27 | P0_12 / RIIC0SCL / DPIN13 / PWGA45O / TAUB0I10 / TAUB0O10 / CSIG0SI / RLIN26TX |
| 28 | P0_13 / RLIN32RX / INTP12 / PWGA46O / TAUB0I12 / TAUB0O12 / CSIG0SO / CAN5RX / INTP5 |
| 29 | P0_14 / RLIN32TX / PWGA47O / TAUB0I14 / TAUB0O14 / CSIG0SC / CAN5TX |
| 30 | P1_0 / RLIN33RX / INTP13 |
| 31 | P1_1 / RLIN33TX |
| 32 | P1_2 / CAN3RX / INTP3 |
| 33 | P1_3 / CAN3TX / DPIN23 |
| 34 | P1_12 / CAN4RX / INTP4 |
| 35 | P1_13 / CAN4TX |
| 36 | P2_6 / ADCA0SEL2 |
| 37 | EVSS |
| 38 | P8_2 / TAUJ0I0 / TAUJ0O0 / DPIN2 / CSIH0CSS0 / INTP6 / PWGA22O / ADCA0I4S |
| 39 | P8_10 / CSIH3CSS3 / DPIN14 / PWGA42O / ADCA0I17S |
| 40 | P8_11 / TAUJ1I2 / TAUJ1O2 / DPIN15 / PWGA43O / CSIH1CSS4 / ADCA0I18S |
| 41 | P8_12 / TAUJ1I3 / TAUJ1O3 / DPIN16 / PWGA44O / CSIH1CSS5 / ADCA0I19S |

**Table 2.3 Pin Assignment 176 Pin LQFP (2/5)**

| Pin No. | Pin Name |
|---|---|
| 42 | JP0_5 / NMI / RTCA0OUT / TAUJ0I3 / TAUJ0O3 / $\overline{\text{DCURDY}}$ / LPDCLKOUT |
| 43 | JP0_4 / $\overline{\text{DCUTRST}}$ |
| 44 | JP0_3 / INTP3 / CSCXFOUT / TAUJ0I2 / TAUJ0O2 / DCUTMS |
| 45 | JP0_2 / INTP2 / TAUJ0I1 / TAUJ0O1 / FPCK / DCUTCK / LPDCLK |
| 46 | JP0_1 / INTP1 /TAUJ0I0 / TAUJ0O0 / FPDT / DCUTDO / LPDO |
| 47 | JP0_0 / INTP0 / FPDR / FPDT / DCUTDI / LPDI / LPDIO |
| 48 | P2_1 / RLIN27TX / CAN6TX |
| 49 | P2_0 / RLIN27RX / INTP6 / CAN6RX |
| 50 | P1_11 / ADCA1TRG2 / RLIN24TX / DPIN22 |
| 51 | P1_10 / RLIN24RX / DPIN21 |
| 52 | P1_9 / RLIN34TX / DPIN20 |
| 53 | P1_8 / RLIN34RX / INTP14 |
| 54 | $\overline{\text{RESET}}$ |
| 55 | EVCC |
| 56 | XT1 |
| 57 | IP0_0 / XT2 |
| 58 | AWOVSS |
| 59 | AWOVCL |
| 60 | REGVCC |
| 61 | X2 |
| 62 | X1 |
| 63 | FLMD0 |
| 64 | P2_3 / RLIN28TX |
| 65 | P2_2 / RLIN28RX |
| 66 | JP0_6 / $\overline{\text{EVTO}}$ |
| 67 | P0_10 / INTP3 / CSIH1CSS1 / DPIN11 / RLIN22TX / TAUB0I6 / TAUB0O6 / CAN4TX |
| 68 | P0_9 / INTP12 / CSIH1CSS0 / DPIN7 / RLIN22RX / TAUB0I4 / TAUB0O4 / CAN4RX / INTP4 |
| 69 | P0_8/ RLIN21TX / DPIN6 / CSIH0CSS6 / $\overline{\text{CSIH1SSI}}$ / TAUB0I2 / TAUB0O2 / CAN3TX |
| 70 | P0_7 / RLIN21RX / DPIN5 / CSCXFOUT / CSIH1RYI / CSIH1RYO / TAUB0I0 / TAUB0O0 / CAN3RX / INTP3 |
| 71 | EVSS |
| 72 | P1_7 / ADCA1TRG1 / RLIN25TX / DPIN19 |
| 73 | P1_6 / RLIN25RX / DPIN18 |
| 74 | P1_5 / ADCA1TRG0 / RLIN35TX / DPIN17 |
| 75 | P1_4 / RLIN35RX / INTP15 |
| 76 | P2_4 / RLIN29RX / ADCA0SEL0 |
| 77 | P2_5 / RLIN29TX / ADCA0SEL1 |
| 78 | P1_14 / RLIN23RX |
| 79 | P1_15 / RLIN23TX |
| 80 | P8_0 / TAUJ0I0 / TAUJ0O0 / DPIN2 / PWGA14O / INTP4 / CSIH0CSS0 / ADCA0I0S |
| 81 | P8_1 / TAPA0ESO / TAUJ0O1 / DPIN0 / PWGA15O / INTP5 / CSIH1CSS3 / ADCA0I1S |
| 82 | P8_3 / TAUJ0I1 / TAUJ0O1 / DPIN3 / CSIH0CSS1 / INTP7 / PWGA23O / ADCA0I5S |

**Table 2.3 Pin Assignment 176 Pin LQFP (3/5)**

| Pin No. | Pin Name |
|---|---|
| 83 | P8_4 / TAUJ0I2 / TAUJ0O2 / DPIN4 / CSIH0CSS2 / INTP8 / PWGA36O / ADCA0I6S |
| 84 | P8_5 / TAUJ0I3 / TAUJ0O3 / NMI / CSIH0CSS3 / INTP9 / PWGA37O / ADCA0I7S |
| 85 | P8_6 / NMI / CSIH0CSS4 / PWGA38O / RTCA0OUT / ADCA0I8S / $\overline{\text{RESETOUT}}$ |
| 86 | P8_7 / CSIH3CSS0 / PWGA39O / ADCA0SEL0 / RTCA0OUT / ADCA0I14S |
| 87 | P8_8 / CSIH3CSS1 / PWGA40O / ADCA0SEL1 / ADCA0I15S |
| 88 | P8_9 / CSIH3CSS2 / PWGA41O / ADCA0SEL2 / ADCA0I16S |
| 89 | A0VSS |
| 90 | A0VREF |
| 91 | AP0_15 / ADCA0I15 |
| 92 | AP0_14 / ADCA0I14 |
| 93 | AP0_13 / ADCA0I13 |
| 94 | AP0_12 / ADCA0I12 |
| 95 | AP0_11 / ADCA0I11 |
| 96 | AP0_10 / ADCA0I10 |
| 97 | AP0_9 / ADCA0I9 |
| 98 | AP0_8 / ADCA0I8 |
| 99 | AP0_7 / ADCA0I7 |
| 100 | AP0_6 / ADCA0I6 |
| 101 | AP0_5 / ADCA0I5 |
| 102 | AP0_4 / ADCA0I4 |
| 103 | AP0_3 / ADCA0I3 |
| 104 | AP0_2 / ADCA0I2 |
| 105 | AP0_1 / ADCA0I1 |
| 106 | AP0_0 / ADCA0I0 |
| 107 | EVSS |
| 108 | P9_0 / NMI / PWGA8O / TAUD0I0 / TAUD0O0 / ADCA0TRG0 / CSIH2CSS0 / KR0I4 / ADCA0I2S |
| 109 | P9_1 / INTP11 / PWGA9O / TAUD0I2 / TAUD0O2 / KR0I5 / CSIH2CSS1 / ADCA0I3S |
| 110 | P9_2 / KR0I6 / PWGA20O / TAPA0ESO / CSIH2CSS2 / ADCA0I9S |
| 111 | P9_3 / KR0I7 / PWGA21O / CSIH2CSS3 / TAUJ1I1 / TAUJ1O1 / ADCA0I10S |
| 112 | P9_4 / CSIH0CSS5 / PWGA33O / TAUJ1I0 / TAUJ1O0 / ADCA0I11S |
| 113 | P9_5 / CSIH0CSS6 / PWGA34O / TAUJ1I1 / TAUJ1O1 / ADCA0I12S |
| 114 | P9_6 / CSIH0CSS7 / PWGA35O / ADCA0I13S |
| 115 | P20_3 / CAN4TX / PWGA67O / RLIN29TX |
| 116 | P20_2 / CAN4RX / INTP4 / PWGA66O / RLIN29RX |
| 117 | P20_1 / RLIN26TX / PWGA65O / CAN6TX |
| 118 | P20_0 / RLIN26RX / PWGA64O / INTP6 / CAN6RX |
| 119 | P20_5 / RLIN23TX / PWGA60O |
| 120 | P20_4 / RLIN23RX / PWGA59O |
| 121 | EVCC |
| 122 | AP1_11 / ADCA1I11 |
| 123 | AP1_10 / ADCA1I10 |

**Table 2.3 Pin Assignment 176 Pin LQFP (4/5)**

| Pin No. | Pin Name |
|---|---|
| 124 | AP1_9 / ADCA1I9 |
| 125 | AP1_8 / ADCA1I8 |
| 126 | AP1_7 / ADCA1I7 |
| 127 | AP1_6 / ADCA1I6 |
| 128 | AP1_5 / ADCA1I5 |
| 129 | AP1_4 / ADCA1I4 |
| 130 | AP1_3 / ADCA1I3 |
| 131 | AP1_2 / ADCA1I2 |
| 132 | AP1_1 / ADCA1I1 |
| 133 | AP1_0 / ADCA1I0 |
| 134 | AP1_15 / ADCA1I15 |
| 135 | AP1_14 / ADCA1I14 |
| 136 | AP1_13 / ADCA1I13 |
| 137 | AP1_12 / ADCA1I12 |
| 138 | A1VREF |
| 139 | A1VSS |
| 140 | EVCC |
| 141 | ISOVCL |
| 142 | ISOVSS |
| 143 | P18_0 / CSIG1RYI / CSIG1RYO / PWGA61O / ADCA1I0S |
| 144 | P18_1 / PWGA62O / ADCA1I1S |
| 145 | P18_2 / PWGA63O / ADCA1I2S |
| 146 | P18_3 / PWGA71O / ADCA1I3S |
| 147 | P18_4 / CSIH1CSS4 / ADCA1I4S |
| 148 | P18_5 / CSIH1CSS5 / ADCA1I5S |
| 149 | P18_6 / ADCA1I6S |
| 150 | P18_7 / ADCA1I7S |
| 151 | EVSS |
| 152 | P10_6 / TAUD0I13 / TAUD0O13 / CSIG0SO / ENCA0TIN0 / ADCA0SEL2 / CAN1RX /INTP1 / MODE2 |
| 153 | P10_7 / TAUD0I15 / TAUD0O15 / CSIG0SC / ENCA0TIN1 / PWGA4O / CAN1TX |
| 154 | P10_8 / TAUD0I10 / TAUD0O10 / CSIG0SI / ENCA0EC / PWGA5O / FLMD1 |
| 155 | P10_9 / TAUD0I12 / TAUD0O12 / RLIN30RX / INTP10 / ENCA0E0 / PWGA6O / CSIH0RYI / CSIH0RYO |
| 156 | P10_10 / TAUD0I14 / TAUD0O14 / RLIN30TX / ENCA0E1 / PWGA7O / CSIH0CSS1 |
| 157 | P10_11 / PWGA16O / RLIN31RX / INTP11 / CSIH1CSS0 / TAUB0I1 / TAUB0O1 |
| 158 | P10_12 / PWGA17O / RLIN31TX / CSIH1CSS1 / TAUB0I3 / TAUB0O3 |
| 159 | P10_13 / $\overline{\text{CSIH0SSI}}$ / PWGA18O / RLIN32RX / INTP12 / TAUB0I5 / TAUB0O5 |
| 160 | P10_14 / ADCA1TRG0 / PWGA19O / RLIN32TX / $\overline{\text{CSIH3SSI}}$ / TAUB0I7 / TAUB0O7 |
| 161 | P11_1 / $\overline{\text{CSIH2SSI}}$ / RLIN20RX / CSIH0CSS7 / PWGA26O / TAUB0I13 / TAUB0O13 |
| 162 | P11_2 / CSIH2SO / RLIN32RX / INTP12 / RLIN20TX / PWGA27O / TAUB0I15 / TAUB0O15 |
| 163 | P11_3 / CSIH2SC / CAN3RX / INTP3 / PWGA28O / TAUB1I1 / TAUB1O1 / RLIN32TX |
| 164 | P11_4 / CSIH2SI / CAN3TX / PWGA29O / TAUB1I3 / TAUB1O3 |

**Table 2.3 Pin Assignment 176 Pin LQFP (5/5)**

| Pin No. | Pin Name |
|---|---|
| 165 | P11_5 / CAN5RX / INTP5 / RLIN33TX / PWGA30O / CSIH3SI / TAUB1I5 / TAUB1O5 |
| 166 | P11_6 / RLIN33RX / INTP13 / CAN5TX / ADCA1TRG1 / PWGA31O / CSIH3SO / TAUB1I7 / TAUB1O7 |
| 167 | P11_7 / INTP5 / PWGA32O / CSIH3SC / TAUB1I9 / TAUB1O9 |
| 168 | P11_15 / CAN2RX / INTP2 / CSIH2CSS4 / PWGA55O / TAUB1I8 / TAUB1O8 |
| 169 | P12_0 / CAN2TX / PWGA56O / TAUB1I10 / TAUB1O10 |
| 170 | P12_1 / RLIN34RX / INTP14 / CSIH2CSS5 / PWGA57O / TAUB1I12 / TAUB1O12 |
| 171 | P12_2 / RLIN34TX / PWGA58O / TAUB1I14 / TAUB1O14 |
| 172 | EVCC |
| 173 | EVSS |
| 174 | P10_0 / TAUD0I1 / TAUD0O1 / CAN0RX /INTP0 / CSCXFOUT / PWGA0O / TAPA0UP / CSIH1SI |
| 175 | P10_1 / TAUD0I3 / TAUD0O3 / CAN0TX / PWGA1O / TAPA0UN / CSIH1SC / MODE0 |
| 176 | P10_2 / TAUD0I5 / TAUD0O5 / RIIC0SDA / KR0I0 / PWGA2O / ADCA0TRG0 / TAPA0VP / CSIH1SO / MODE1 |

## 2.2 Pin Description

Table 2.4 Pin Functions (1/4)

| Pin Name | No. of Pins | | | IO | Pin Function | Unit |
|---|---|---|---|---|---|---|
| | 100 pins | 144 pins | 176 pins | | | |
| AnVREF | √ | √ | √ | — | ADCAn voltage supply and reference voltage | ADCAn |
| | n = 0 | n = 0, 1 | n = 0, 1 | | | |
| AnVSS | √ | √ | √ | — | ADCAn ground | |
| | n = 0 | n = 0, 1 | n = 0, 1 | | | |
| ADCA0Im | √ | √ | √ | I | ADCA0 input channel m with 12-bit resolution | |
| | m = 0 to 15 | m = 0 to 15 | m = 0 to 15 | | | |
| ADCA1Im | — | √ | √ | I | ADCA1 input channel m with 12-bit resolution | |
| | | m = 0 to 7 | m = 0 to 15 | | | |
| ADCA0ImS | √ | √ | √ | I | ADCA0 input channel m with 10-bit resolution | |
| | m = 0 to 19 | m = 0 to 19 | m = 0 to 19 | | | |
| ADCA1ImS | — | √ | √ | I | ADCA1 input channel m with 10-bit resolution | |
| | | m = 0 to 3 | m = 0 to 7 | | | |
| ADCA0SELy | √ | √ | √ | O | External MPX selection pin y for ADCA0 input | |
| | y = 0 to 2 | y = 0 to 2 | y = 0 to 2 | | | |
| ADCAnTRGy | √ | √ | √ | I | ADCAn external trigger pin y | |
| | n = 0,<br>y = 0 to 2 | n = 0, 1,<br>y = 0 to 2 | n = 0, 1,<br>y = 0 to 2 | | | |
| AP0_m | √ | √ | √ | IO | Analog port 0_m | Port |
| | m = 0 to 15 | m = 0 to 15 | m = 0 to 15 | | | |
| AP1_m | — | √ | √ | IO | Analog port 1_m | |
| | | m = 0 to 7 | m = 0 to 15 | | | |
| APO | √ | √ | √ | O | Port output signal for analog input | LPS |
| AWOVCL | √ | √ | √ | — | Voltage regulator for Always-On area (AWO area) capacitor connection | Power |
| AWOVSS | √ | √ | √ | — | Internal logic for Always-On area (AWO area) ground | |
| CANmRX | √ | √ | √ | I | CANm receive data input | RS-CANn |
| | m = 0 to 5 | m = 0 to 5 | m = 0 to 6 | | | |
| CANmTX | √ | √ | √ | O | CANm transmit data output | |
| | m = 0 to 5 | m = 0 to 5 | m = 0 to 6 | | | |
| CSCXFOUT | √ | √ | √ | O | Clock output | Clock |
| CSIGnRYI | √ | √ | √ | I | CSIGn ready (1) / busy (0) input signal | CSIGn |
| | n = 0 | n = 0, 1 | n = 0, 1 | | | |
| CSIGnRYO | √ | √ | √ | O | CSIGn ready (1) / busy (0) output signal | |
| | n = 0 | n = 0, 1 | n = 0, 1 | | | |
| CSIGnSC | √ | √ | √ | IO | CSIGn serial clock signal | |
| | n = 0 | n = 0, 1 | n = 0, 1 | | | |
| CSIGnSI | √ | √ | √ | I | CSIGn serial data input | |
| | n = 0 | n = 0, 1 | n = 0, 1 | | | |
| CSIGnSO | √ | √ | √ | O | CSIGn serial data output | |
| | n = 0 | n = 0, 1 | n = 0, 1 | | | |
| $\overline{\text{CSIGnSSI}}$ | √ | √ | √ | I | CSIGn SS function control input signal | |
| | n = 0 | n = 0, 1 | n = 0, 1 | | | |

Table 2.4 Pin Functions (2/4)

| Pin Name | No. of Pins | | | IO | Pin Function | Unit |
|---|---|---|---|---|---|---|
| | 100 pins | 144 pins | 176 pins | | | |
| CSIHnCSS0 | √ | √ | √ | O | CSIHn serial peripheral chip select signal 0 | CSIHn |
| | n = 0 to 3 | n = 0 to 3 | n = 0 to 3 | | | |
| CSIHnCSS1 | √ | √ | √ | O | CSIHn serial peripheral chip select signal 1 | |
| | n = 0 to 3 | n = 0 to 3 | n = 0 to 3 | | | |
| CSIHnCSS2 | √ | √ | √ | O | CSIHn serial peripheral chip select signal 2 | |
| | n = 0 to 3 | n = 0 to 3 | n = 0 to 3 | | | |
| CSIHnCSS3 | √ | √ | √ | O | CSIHn serial peripheral chip select signal 3 | |
| | n = 0 to 3 | n = 0 to 3 | n = 0 to 3 | | | |
| CSIHnCSS4 | √ | √ | √ | O | CSIHn serial peripheral chip select signal 4 | |
| | n = 0, 1 | n = 0 to 2 | n = 0 to 2 | | | |
| CSIHnCSS5 | √ | √ | √ | O | CSIHn serial peripheral chip select signal 5 | |
| | n = 0, 1 | n = 0 to 2 | n = 0 to 2 | | | |
| CSIHnCSS6 | √ | √ | √ | O | CSIHn serial peripheral chip select signal 6 | |
| | n = 0 | n = 0 | n = 0 | | | |
| CSIHnCSS7 | √ | √ | √ | O | CSIHn serial peripheral chip select signal 7 | |
| | n = 0 | n = 0 | n = 0 | | | |
| CSIHnRYI | √ | √ | √ | I | CSIHn ready (1) / busy (0) input signal | |
| | n = 0 to 3 | n = 0 to 3 | n = 0 to 3 | | | |
| CSIHnRYO | √ | √ | √ | O | CSIHn ready (1) / busy (0) output signal | |
| | n = 0 to 3 | n = 0 to 3 | n = 0 to 3 | | | |
| CSIHnSC | √ | √ | √ | IO | CSIHn serial clock signal | |
| | n = 0 to 3 | n = 0 to 3 | n = 0 to 3 | | | |
| CSIHnSI | √ | √ | √ | I | CSIHn serial data input | |
| | n = 0 to 3 | n = 0 to 3 | n = 0 to 3 | | | |
| CSIHnSO | √ | √ | √ | O | CSIHn serial data output | |
| | n = 0 to 3 | n = 0 to 3 | n = 0 to 3 | | | |
| $\overline{\text{CSIHnSSI}}$ | √ | √ | √ | I | CSIHn slave select input signal | |
| | n = 0 to 3 | n = 0 to 3 | n = 0 to 3 | | | |
| $\overline{\text{DCURDY}}$ | √ | √ | √ | O | Debug ready | OCD |
| DCUTCK | √ | √ | √ | I | Debug clock | |
| DCUTDI | √ | √ | √ | I | Debug data input | |
| DCUTDO | √ | √ | √ | O | Debug data output | |
| DCUTMS | √ | √ | √ | I | Debug mode select | |
| $\overline{\text{DCUTRST}}$ | √ | √ | √ | I | Debug reset | |
| DPINm | √ | √ | √ | I | Digital port input m | LPS |
| | m = 0 to 16 | m = 0 to 23 | m = 0 to 23 | | | |
| DPO | √ | √ | √ | O | Port output signal for digital input | |
| ENCA0TINm | √ | √ | √ | I | ENCA0 capture trigger input m | ENCA |
| | m = 0, 1 | m = 0, 1 | m = 0, 1 | | | |
| ENCA0EC | √ | √ | √ | I | ENCA0 encoder clear input | |
| ENCA0E0 | √ | √ | √ | I | ENCA0 encoder input 0 | |
| ENCA0E1 | √ | √ | √ | I | ENCA0 encoder input 1 | |
| EVCC | √ | √ | √ | — | Port buffer voltage supply | Power |
| EVSS | √ | √ | √ | — | Port buffer ground | |
| $\overline{\text{EVTO}}$ | — | √ | √ | O | Event output | TEU_OUT |
| FLMD0 | √ | √ | √ | I | Operating mode select pin 0 | Mode |
| FLMD1 | √ | √ | √ | I | Operating mode select pin 1 | |

Table 2.4 Pin Functions (3/4)

| Pin Name | No. of Pins | | | IO | Pin Function | Unit |
|---|---|---|---|---|---|---|
| | 100 pins | 144 pins | 176 pins | | | |
| FPDR | √ | √ | √ | I | Serial Communication Interface RXD | FLASH |
| FPDT | √ | √ | √ | O | Serial Communication Interface TXD | |
| FPCK | √ | √ | √ | I | Serial Communication Interface clock | |
| INTPm | √ | √ | √ | I | External interrupt input m | INTC |
| | m = 0 to 8, 10 to 13 | m = 0 to 15 | m = 0 to 15 | | | |
| IP0_0 | — | √ | √ | I | Input port 0_0 | Port |
| ISOVCL | √ | √ | √ | — | Voltage regulator for Isolated area (ISO area) capacitor connection | Power |
| ISOVSS | √ | √ | √ | — | Internal logic for Isolated area (ISO area) area ground | |
| JP0_m | √ | √ | √ | IO | JTAG port 0_m | Port |
| | m = 0 to 5 | m = 0 to 6 | m = 0 to 6 | | | |
| KR0Im | √ | √ | √ | I | KR0 key input signal | KR0 |
| | m = 0 to 7 | m = 0 to 7 | m = 0 to 7 | | | |
| LPDCLK | √ | √ | √ | I | LPD clock input (4-pin mode) | LPD |
| LPDCLKOUT | √ | √ | √ | O | LPD clock output (4-pin mode) | |
| LPDI | √ | √ | √ | I | LPD data input (4-pin mode) | |
| LPDIO | √ | √ | √ | IO | LPD data input / output (1-pin mode) | Port |
| LPDO | √ | √ | √ | O | LPD data output (4-pin mode) | |
| MODEm | √ | √ | √ | I | Sub operating mode select (Boundary scan) | Mode |
| | m = 0 to 2 | m = 0 to 2 | m = 0 to 2 | | | |
| NMI | √ | √ | √ | I | External non-maskable interrupt input | INTC |
| P0_m | √ | √ | √ | IO | Port 0_m | Port |
| | m = 0 to 14 | m = 0 to 14 | m = 0 to 14 | | | |
| P1_m | — | √ | √ | IO | Port 1_m | |
| | | m = 0 to 11 | m = 0 to 15 | | | |
| P2_m | — | — | √ | IO | Port 2_m | |
| | | | m = 0 to 6 | | | |
| P8_m | √ | √ | √ | IO | Port 8_m | |
| | m = 0 to 12 | m = 0 to 12 | m = 0 to 12 | | | |
| P9_m | √ | √ | √ | IO | Port 9_m | |
| | m = 0 to 6 | m = 0 to 6 | m = 0 to 6 | | | |
| P10_m | √ | √ | √ | IO | Port 10_m | |
| | m = 0 to 15 | m = 0 to 15 | m = 0 to 15 | | | |
| P11_m | √ | √ | √ | IO | Port 11_m | |
| | m = 0 to 7 | m = 0 to 15 | m = 0 to 15 | | | |
| P12_m | — | √ | √ | IO | Port 12_m | |
| | | m = 0 to 2 | m = 0 to 5 | | | |
| P18_m | — | √ | √ | IO | Port 18_m | |
| | | m = 0 to 3 | m = 0 to 7 | | | |
| P20_m | — | √ | √ | IO | Port 20_m | |
| | | m = 4, 5 | m = 0 to 5 | | | |
| PWGAnO | √ | √ | √ | O | PWGAn output signal | PWM-Diag |
| | n = 0 to 47 | n = 0 to 63 | n = 0 to 71 | | | |
| REGVCC | √ | √ | √ | — | Voltage regulators voltage supply | Power |
| $\overline{\text{RESET}}$ | √ | √ | √ | I | External reset input | Reset |
| $\overline{\text{RESETOUT}}$ | √ | √ | √ | O | Reset output | |
| RIIC0SCL | √ | √ | √ | IO | RIIC0 serial clock | RIIC0 |
| RIIC0SDA | √ | √ | √ | IO | RIIC0 serial data | |

Table 2.4 Pin Functions (4/4)

| Pin Name | No. of Pins | | | IO | Pin Function | Unit |
|---|---|---|---|---|---|---|
| | 100 pins | 144 pins | 176 pins | | | |
| RLIN2mRX | √ | √ | √ | I | RLIN2m receive data input | RLIN2m |
| | m = 0 to 2 | m = 0 to 5 | m = 0 to 9 | | | |
| RLIN2mTX | √ | √ | √ | O | RLIN2m transmit data output | |
| | m = 0 to 2 | m = 0 to 5 | m = 0 to 9 | | | |
| RLIN3nRX | √ | √ | √ | I | RLIN3n receive data input | RLIN3n |
| | n = 0 to 3 | n = 0 to 5 | n = 0 to 5 | | | |
| RLIN3nTX | √ | √ | √ | O | RLIN3n transmit data output | |
| | n = 0 to 3 | n = 0 to 5 | n = 0 to 5 | | | |
| RTCA0OUT | — | √ | √ | O | RTCA0 1Hz output | RTCA0 |
| SELDPk | √ | √ | √ | O | External multiplexer selection output signal k for digital port | LPS |
| | k = 0 to 2 | k = 0 to 2 | k = 0 to 2 | | | |
| TAPA0ESO | √ | √ | √ | I | Hi-Z control | Motor control |
| TAPA0UN | √ | √ | √ | O | Motor control output U phase (negative) | |
| TAPA0UP | √ | √ | √ | O | Motor control output U phase (positive) | |
| TAPA0VN | √ | √ | √ | O | Motor control output V phase (negative) | |
| TAPA0VP | √ | √ | √ | O | Motor control output V phase (positive) | |
| TAPA0WN | √ | √ | √ | O | Motor control output W phase (negative) | |
| TAPA0WP | √ | √ | √ | O | Motor control output W phase (positive) | |
| TAUD0Im | √ | √ | √ | I | TAUD0 channel input m | TAUD0 |
| | m = 0 to 15 | m = 0 to 15 | m = 0 to 15 | | | |
| TAUD0Om | √ | √ | √ | O | TAUD0 channel output m | |
| | m = 0 to 15 | m = 0 to 15 | m = 0 to 15 | | | |
| TAUBnIm | √ | √ | √ | I | TAUBn channel input m | TAUBn |
| | n = 0,<br>m = 0 to 15 | n = 0,<br>m = 0 to 15 | n = 0, 1,<br>m = 0 to 15 | | | |
| TAUBnOm | √ | √ | √ | O | TAUBn channel output m | |
| | n = 0,<br>m = 0 to 15 | n = 0,<br>m = 0 to 15 | n = 0, 1,<br>m = 0 to 15 | | | |
| TAUJnIm | √ | √ | √ | I | TAUJn channel input m | TAUJn |
| | n = 0, 1,<br>m = 0 to 3 | n = 0, 1,<br>m = 0 to 3 | n = 0, 1,<br>m = 0 to 3 | | | |
| TAUJnOm | √ | √ | √ | O | TAUJn channel output m | |
| | n = 0, 1,<br>m = 0 to 3 | n = 0, 1,<br>m = 0 to 3 | n = 0, 1,<br>m = 0 to 3 | | | |
| X1, X2 | √ | √ | √ | — | MainOSC connections | MOSC |
| XT1, XT2 | — | √ | √ | — | SubOSC connections | SOSC |

**CAUTION**

- **When pin functions for a peripheral module are allocated to multiple pins, use the pins from the same port group or nearby pins as the pins for a given channel.**
  **(e.g.) When RS-CAN channel 0 is used:**
  **CAN0TX P0_0 P10_1**
  **CAN0RX P0_1 P10_0**
  **Use one of the following pin combinations:**
  - **P0_0 and P0_1, or**
  - **P10_0 and P10_1.**
- **The combinations of P0_0 and P10_0, and P0_1 and P10_1 are not allowed.**

## 2.3 Pin Functions During and After Reset

**Table 2.5 Pin Functions During and After Reset**

| Pins | During Reset | After Reset |
|---|---|---|
| JP0_0 | High impedance | JP0_0 : Input<br>Serial programming mode: FPDR, FPDT (1 wire UART)<br>FPDR (2 wire UART)<br>Nexus I/F : DCUTDI input<br>LPD (4pin) : LPDI input<br>LPD (1pin) : LPDIO input/output |
| JP0_1 | High impedance | JP0_1 : Input<br>Serial programming mode: FPDT<br>Nexus I/F : DCUTDO output<br>LPD (4pin) : LPDO output<br>LPD (1pin) : High impedance |
| JP0_2 | High impedance | JP0_2 : Input<br>Serial programming mode: FPCK<br>Nexus I/F : DCUTCK input<br>LPD (4pin) : LPDCLK input<br>LPD (1pin) : High impedance |
| JP0_3 | High impedance | JP0_3 : Input<br>Serial programming mode: High impedance<br>Nexus I/F : DCUTMS input<br>LPD (4pin) : High impedance<br>LPD (1pin) : High impedance |
| JP0_4 | Input*3 | JP0_4 : Input<br>Serial programming mode: High impedance<br>Nexus I/F : $\overline{\text{DCUTRST}}$ input*1<br>LPD (4pin) : High impedance<br>LPD (1pin) : High impedance |
| JP0_5 | High impedance | JP0_5 : Input<br>Serial programming mode: High impedance<br>Nexus I/F : $\overline{\text{DCURDY}}$ output<br>LPD (4pin) : LPDCLKOUT output<br>LPD (1pin) : High impedance |
| JP0_6 | High impedance | JP0_6 : Input<br>Serial programming mode: High impedance<br>Nexus I/F : $\overline{\text{EVTO}}$ output<br>LPD (4pin) : High impedance<br>LPD (1pin) : High impedance |
| P8_6 | Output*2, *4 | Output (OPBT0.RESETOUTEN = 1)*2<br>High impedance (OPBT0.RESETOUTEN = 0)*2, *4 |
| P0 to P2, P8 to P12, P18 and P20 (except P8_6, P10_1, P10_2, P10_6 and P10_8) | High impedance | High impedance |
| P10_1 | High impedance | High impedance (FLMD0 = 0)<br>High impedance (FLMD0 = 1, FLMD1 = 0)<br>MODE0 input (FLMD0 = 1, FLMD1 = 1) |
| P10_2 | High impedance | High impedance (FLMD0 = 0)<br>High impedance (FLMD0 = 1, FLMD1 = 0)<br>MODE1 input (FLMD0 = 1, FLMD1 = 1) |
| P10_6 | High impedance | High impedance (FLMD0 = 0)<br>High impedance (FLMD0 = 1, FLMD1 = 0)<br>High impedance (FLMD0 = 1, FLMD1 = 1, MODE0 = 0, MODE1 = 0)<br>High impedance (FLMD0 = 1, FLMD1 = 1, MODE0 = 0, MODE1 = 1)<br>High impedance (FLMD0 = 1, FLMD1 = 1, MODE0 = 1, MODE1 = 0)<br>MODE2 input (FLMD0 = 1, FLMD1 = 1, MODE0 = 1, MODE1 = 1) |
| P10_8 | High impedance | High impedance (FLMD0 = 0)<br>FLMD1 input (FLMD0 = 1) |
| FLMD0 | Input | Input |
| $\overline{\text{RESET}}$ | Input | Input |
| AP0, AP1 | High impedance | High impedance |

Note 1. When Nexus is enabled and no external device is connected, the level of the pin must always be fixed to low.

Note 2. $\overline{\text{RESETOUT}}$ is output. For details, see **Section 2.11, Port (Special I/O) Function Overview**.

Note 3. When the power is turned on or when a reset is deasserted, the JP0_4 pin should be driven Low.

Note 4. If OPBT0.RESETOUTEN = 0, P8_6 pin status has a possibility to become unstable (less than 15 µs) at the transition moment to reset status by internal reset factors.

## 2.4 Port state in Standby Mode

For the port state in standby mode, see **Section 14.1.4, I/O Buffer Control**.

## 2.5 Recommended Connection of Unused Pins

If the pins are not used, it is recommended to connect them as shown below.

**Table 2.6 Recommended Connection of Unused Pins**

| Pin | Recommended Connection of Unused Pins |
|---|---|
| A0VREF, A1VREF*1 | Connected to EVCC |
| A0VSS, A1VSS*1 | Connected to EVSS |
| $\overline{\text{RESET}}$ | Connected to EVCC via a resistor |
| XT1 | Connected to REGVCC or AWOVSS via a resistor*3 (bit 0 of IPIBC0 = 1)<br>Connected to AWOVSS (bit 0 of IPIBC0 = 0) |
| X1 | Connected to AWOVSS via a resistor |
| X2 | Open |
| FLMD0 | Must be used |
| IP0_0 | Connected to REGVCC or AWOVSS via a resistor*3 (bit 0 of IPIBC0 = 1)<br>Open (bit 0 of IPIBC0 = 0) |
| JP0 (excluding JP0_4)<br>P0<br>P1<br>P2<br>P8 (excluding P8_6)<br>P9<br>P20 | Input: Open (when the PIBCn_m and PMCn_m bits are 0)<br>Connected to EVCC or EVSS via a resistor (when the PIBCn_m or PMCn_m bits are 1)<br>Output: Open |
| P8_6 | Input: Open (when the PIBCn_m and PMCn_m bits are 0)<br>Connected to EVSS via a resistor (when the PIBCn_m or PMCn_m bits are 1)<br>Output: Open |
| JP0_4 | Connected to EVSS via a resistor*2 |
| P10 (excluding P10_1, P10_2, P10_6, P10_8)<br>P11<br>P12<br>P18 | Input: Open (when the PIBCn_m and PMCn_m bits are 0)<br>Connected to EVCC or EVSS via a resistor (when the PIBCn_m or PMCn_m bits are 1)<br>Output: Open |
| P10_1, P10_2, P10_6, P10_8 | Input: Open (when the PIBCn_m and PMCn_m bits are 0)<br>Connected to EVSS via a resistor (when the PIBCn_m or PMCn_m bits are 1)<br>Output: Open |
| AP0 | Input: Open (when the PIBCn_m bit is 0)<br>Connected to A0VREF or A0VSS via a resistor (when the PIBCn_m bit is 1)<br>Output: Open |
| AP1 | Input: Open (when the PIBCn_m bit is 0)<br>Connected to A1VREF or A1VSS via a resistor (when the PIBCn_m bit is 1)<br>Output: Open |
| Nexus/LPD I/F (JP0) | DCUTDI/LPDI/LPDIO (JP0_0): Connected to EVCC via a resistor<br>DCUTDO/LPDO (JP0_1): Open<br>DCUTCK/LPDCLK (JP0_2): Open<br>DCUTMS (JP0_3): Connected to EVCC via a resistor<br>$\overline{\text{DCUTRST}}$ (JP0_4): Connected to EVSS via a resistor*2<br>$\overline{\text{DCURDY}}$/LPDCLKOUT (JP0_5):Open<br>$\overline{\text{EVTO}}$ (JP0_6): Open*1 |

Note 1. Only in 176 and 144 pin devices.

Note 2. For details, see the specifications of the development tool.

Note 3. XT1 = IP0_0 (XT2) = REGVCC or AWOVSS should be set.
XT1 is connected to IP0_0 (XT2) through an internal resistor. Therefore, it is necessary to maintain equal voltage level in order not to make a current path.

# 2.6 RH850/F1K Port Features

## 2.6.1 Port Group

The RH850/F1K provides the following port groups, indicated by the numbers in the table below.

**Table 2.7 Port Groups in RH850/F1K**

| No. of Pins | Port Group | RH850/F1K |
|---|---|---|
| 100 pins | Number | 7 |
| | Name | P0, P8 to P11, JP0, AP0 |
| 144 pins | Number | 13 |
| | Name | P0, P1, P8 to P12, P18, P20, JP0, AP0, AP1, IP0 |
| 176 pins | Number | 14 |
| | Name | P0 to P2, P8 to P12, P18, P20, JP0, AP0, AP1, IP0 |

## 2.6.2 Port Group Index n

Throughout this section, the port groups are identified by using the index “n” (n = 0 to 2, 8 to 12, 18, and 20). For example, the port mode control register of the Pn pin is PMCn.

## 2.6.3 Register Base Address

Port and JTAG port base addresses are listed in the following table.

Port and JTAG port register addresses are given as offsets from the base addresses.

**Table 2.8 Register Base Addresses**

| Base Address Name | Base Address |
|---|---|
| <PORTn_base> | FFC1 $0000_H$ |
| <JPORT0_base> | FFC2 $0000_H$ |

## 2.6.4 Clock Supply

The clock supply to ports is shown in the following table.

**Table 2.9 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name |
|---|---|---|
| Port | Register access clock | CPUCLK4 |
| | | CPUCLK2 |

# 2.7 Port Functions

This product has various pins for input/output ports. The ports are organized in port groups.

The RH850/F1K also has several control registers to enable pins to be used as other than general purpose input/output pins.

For a description of the terms pin, port, and port group, see **Section 2.7.2, Terms**.

## 2.7.1 Functional Overview

- All the port settings can be specified individually.
- The maximum number of bits (pins) in a port is 16.
- The output level of any pin can be set independently without affecting the other pins in the same port.
- Input buffers are enabled through registers settings.
- Pin level is read by dedicated port-pin-read register (PPR)
- All possible port functions are shown in the tables listed below.
  **Table 2.38**, **Table 2.40**, **Table 2.42**, **Table 2.44**, **Table 2.46**, **Table 2.48**, **Table 2.50**, **Table 2.52**, **Table 2.54**, **Table 2.56**, **Table 2.58**, **Table 2.60**, **Table 2.62**, **Table 2.64** and **Section 2.9.2, Pin Function Configuration**.

**CAUTION**

**Some input or output functions may be assigned to more than one port. Only activate a given function on a single pin. Do not activate a function on multiple pins at the same time. This also applies in cases where multiple peripheral functions are assigned to a single multiplexed function and only one of these functions is used.**

**[Example]**

**INTP0 is assigned to the following pins on this device. However, the INTP0 function should not be activated on more than one pin. After activating the function on one pin, do not activate it on another.**

- **JP0_0 (1st input alternative function)**
- **P0_1 (2nd, 3rd input alternative function)**
- **P10_0 (2nd input alternative function)**

**In the above case, when the 1st input alternative function (INTP0) of JP0_0 is selected, using the 2nd input alternative function (CAN0RX/INTP0) of P0_1 only for the CAN signal is also prohibited.**

## 2.7.2 Terms

The following terms are used in this section:

**Pin**

Denotes the physical pin. Every pin is denoted by a unique pin number.

A pin can be used in several modes. Each pin is assigned a name that reflects its function, which is determined by the selected mode.

**Port group**

Denotes a group of pins. All the pins of a specific port group are controlled by the same port control register.

**Port mode and ports**

A pin in port mode works as a general purpose input/output pin. It is then called “port”.

The corresponding name is Pn_m. For example, P0_7 denotes port 7 of port group 0. It is referenced as “port P0_7”.

**Alternative mode**

In alternative mode, a pin can be used for various non-general-purpose input/output functions, such as the input/output pin of on-chip peripherals.

The corresponding pin name depends on the selected function. For example, pin INTP0 denotes the pin for one of the external interrupt inputs.

Note that two different names can refer to the same physical pin, for example P0_0 and INTP0. The different names indicate the function of the pin at that time.

### 2.7.2.1 JTAG Ports

The JTAG port groups are used for connecting a debugger for on-chip debugging. These are special port groups provided because the microcontroller cannot be used for the user's application while on-chip debugging is being executed. When a debugger is not connected and the microcontroller is operating normally, these port groups can be used in the same way as the other port groups.

JTAG port group registers and bit names are prefixed by a “J”. For example, JP0 denotes JTAG port group 0, and JPM0.JPM0_m denotes the JPM0_m port mode bit of the JPM0 port mode register.

**NOTE**

In this section, the descriptions about all ports and their registers other than PFCAEn and PIPCn apply to the JTAG port unless otherwise specified.

### 2.7.3 Overview of Pin Functions

Pins can operate in three modes.

- Port mode (PMCn.PMCn_m bit = 0)

  A pin in port mode operates as a general purpose input/output pin. The I/O mode is selected by setting the PMn.PMn_m bit.

- Software I/O control alternative mode (PMCn.PMCn_m bit = 1, PIPCn.PIPCn_m bit = 0)

  In this mode, the pins operate as alternative functions. The I/O mode is selected by setting the PMn.PMn_m bit.

- Direct I/O control alternative mode (PMCn.PMCn_m bit = 1, PIPCn.PIPCn_m bit = 1)

  In this mode, the pins operate as alternative functions. Unlike the software I/O control alternative mode, however, the I/O mode is directly controlled by the alternative function.

An overview of the register settings is given in the tables below.

**Table 2.10 Pin Function Configuration (Overview)**

| Mode | Bit | | | I/O |
|---|---|---|---|---|
| | PMCn_m | PMn_m | PIPCn_m | |
| Port mode | 0 | 0 | X | O |
| | | 1*1 | | I |
| Software I/O control alternative mode | 1 | 0 | 0 | O |
| | | 1 | 0 | I |
| Direct I/O control alternative mode | | X | 1 | Controlled by the alternative function |

Note 1. The input buffer must be enabled (PIBCn_m bit = 1).

- Software I/O control alternative mode (PIPCn.PIPCn_m bit = 0)
  - Output (PMn_m bit = 0): Alternative output mode 1 to Alternative output mode 7
  - Input (PMn_m bit = 1): Alternative input mode 1 to Alternative input mode 7
- Direct I/O control alternative mode (PIPCn.PIPCn_m bit = 1)
  - The I/O mode for Alternative output mode 1 to Alternative output mode 7 and Alternative input mode 1 to Alternative input mode 7 is directly selected by the alternative function.

**Table 2.11 Alternative Mode Selection Overview (PMCn.PMCn_m Bit = 1)**

| Mode | Register | | | | | I/O |
|---|---|---|---|---|---|---|
| | PIPC*1 | PM*1 | PFCAE | PFCE | PFC | |
| Alternative output mode 1 (ALT-OUT1) | 0 | 0 | 0 | 0 | 0 | O |
| Alternative input mode 1 (ALT-IN1) | | 1 | | | | I |
| Alternative output mode 2 (ALT-OUT2) | | 0 | 0 | 0 | 1 | O |
| Alternative input mode 2 (ALT-IN2) | | 1 | | | | I |
| Alternative output mode 3 (ALT-OUT3) | | 0 | 0 | 1 | 0 | O |
| Alternative input mode 3 (ALT-IN3) | | 1 | | | | I |
| Alternative output mode 4 (ALT-OUT4) | | 0 | 0 | 1 | 1 | O |
| Alternative input mode 4 (ALT-IN4) | | 1 | | | | I |
| Alternative output mode 5 (ALT-OUT5) | | 0 | 1 | 0 | 0 | O |
| Alternative input mode 5 (ALT-IN5) | | 1 | | | | I |
| Alternative output mode 6 (ALT-OUT6) | | 0 | 1 | 0 | 1 | O |
| Alternative input mode 6 (ALT-IN6) | | 1 | | | | I |
| Alternative output mode 7 (ALT-OUT7) | | 0 | 1 | 1 | 0 | O |
| Alternative input mode 7 (ALT-IN7) | | 1 | | | | I |
| Other than the above | Setting prohibited | | | | | |

Note 1. If PIPCn.PIPCn_m bit = 1, the I/O direction is directly controlled by the peripheral (alternative) function and PM is ignored.

If a pin is in alternative mode (PMCn.PMCn_m bit = 1), one of up to seven alternative functions can be selected for that pin by using the PFCn, PFCEn, and PFCAEn registers.

## 2.7.4 Pin Data Input/Output

The registers used for data input/output are described below.

The location that is read via the PPRn register differs depending on the pin mode.

### 2.7.4.1 Output Data

In the port mode (PMCn.PMCn_m bit = 0), the value of the Pn.Pn_m bit is output from the Pn_m pin.

### 2.7.4.2 Input Data

When the PPRn register is read, either the value of the Pn_m pin, the value of the Pn.Pn_m bit, or the value output by the alternative function is returned.

Which value is returned depends on the pin mode and setting of several control bits.

The different PPRn read modes are shown in the table below.

**Table 2.12 PPRn_m Read Values**

| PMC n_m | PM n_m | PIBC n_m | PIPC n_m | PODC n_m | Mode | PPRn_m Read Value |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | X | X | Port input, input buffer disabled | Pn.Pn_m bit |
| | | 1 | | X | Port input, input buffer enabled | Pn_m pin |
| | 0 | X | | 0 | Port push-pull output | Pn.Pn_m bit*1 |
| | | | | 1 | Port open-drain output | |
| 1 | 1 | X | 0 | X | Software I/O control alternative input | Pn_m pin |
| | 0 | | | 0 | Software I/O control alternative push-pull output | Output signal from the alternative function*1 |
| | | | | 1 | Software I/O control alternative open-drain output | |
| | X | | 1 | 0 | Direct I/O control alternative input or push-pull output | I/O port in alternative mode:<br>• Input: Pn_m pin<br>• Output: Output signal from the alternative function*1 |
| | | | | 1 | Direct I/O control alternative input or open-drain output | |

Note 1. When PBDCn_m = 1, the level of the Pn_m pin is returned by the PPRn_m bit.

The control registers in the above table have the following effects:

- PMCn.PMCn_m bit
  This bit selects port mode (PMCn_m = 0) or alternative mode (PMCn_m = 1).
- PMn.PMn_m bit
  This bit selects input (PMn_m = 1) or output (PMn_m = 0) when the port mode (PMCn_m = 0) and software I/O control alternative mode (PMCn_m = 1, PIPCn_m = 0) have been selected.
- PIBCn.PIBCn_m bit
  This bit disables (PIBCn_m = 0) or enables (PIBCn_m = 1) the input buffer in input port mode (PMCn_m = 0 and PMn_m = 1). If the input buffer is disabled, PPRn_m reads the Pn.Pn_m bit; otherwise the Pn_m pin level is returned.
  If alternative mode (PMCn_m = 1), this bit must always set to 0.
- PIPCn.PIPCn_m bit
  This bit selects software I/O control alternative mode or direct I/O control alternative mode.

- PODCn.PODCn_m bit
  This bit selects push-pull output (PODCn_m = 0) or open-drain output (PODCn_m = 1).
- PBDCn.PBDCn_m bit
  In output mode, when this bit is set to 1, the pin enters the bidirectional mode. In bidirectional mode, the level of the signal on a Pn_m pin can be read from PPRn.PPRn_m.

**CAUTION**

**When using Pn_m as an alternative output function (PMCn.PMCn_m bit = 1, PMn.PMn_m bit = 0), the level of the Pn_m pin can be read at the PPRn.PPRn_m bit by enabling bidirectional mode (PBDCn.PBDCn_m bit = 1).**

**Note, however, that the level of the Pn_m pin will be input to the alternative input function that the Pn_m pin is being used as.**

### 2.7.4.3 Writing to the Pn Register

The data to be output via port Pn_m in port mode (PMCn.PMCn_m bit = 0) is held in port register Pn.

Pn data can be overwritten in two ways:

- By writing data directly to the Pn register.
  In this case, new data can be written directly to the Pn register.
- By performing an indirect bitwise operation (a “set”, “reset”, or “not” operation) on the Pn register.
  An indirect bitwise operation (“set”, “reset”, or “not”) can be performed on the Pn register by using the following two registers:
  - Port Set/Reset register PSRn
    If the PSRn.PSRn (m + 16) bit = 1, the value of the Pn.Pn_m bit is determined by the value of the PSRn.PSRn_m bit.
    In other words, the Pn_m bit can be set or reset without writing directly to the Pn register.
  - Port NOT register PNOTn
    By setting PNOTn.PNOTn_m bit to 1, the Pn.Pn_m bit can be inverted without writing directly to the Pn register.

An indirect bitwise operation on the Pn register (“set”, “reset”, or “not”) has no effect on the bits that do not need to be updated, allowing you to overwrite only the bit or bits that need to be overwritten.

## 2.8 Schematic View of Port Control

The following figure is a schematic view of the port control functions.

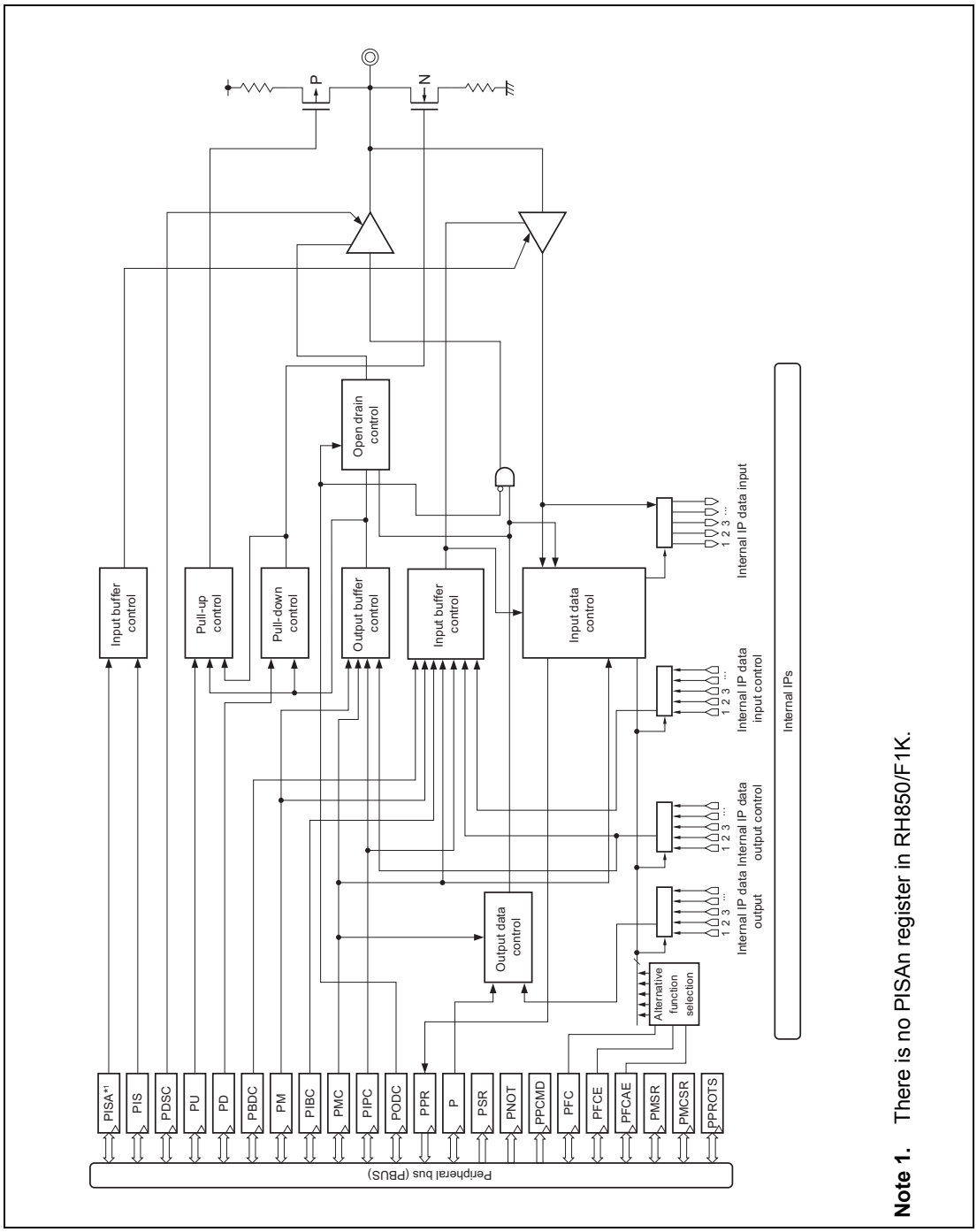



**Figure 2.4 Schematic View of Port Control**

**CAUTION**

**Use documented alternative functions only.**

## 2.9 Port Group Configuration Registers

This section starts with an overview of all configuration registers and then describes all registers in detail. The configuration registers are grouped as follows:

- **Section 2.9.2, Pin Function Configuration**
- **Section 2.9.3, Pin Data Input/Output**
- **Section 2.9.4, Configuration of Electrical Characteristics**

### 2.9.1 Overview

The following registers are used for setting the individual pins of the port groups.

For details on <PORTn_base> and <JPORT0_base>, see **Section 2.6.3, Register Base Address**.

**Table 2.13 Port Group Configuration Registers (1/2)**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| **Pin function configuration** | | | |
| PORT | Port mode control register | PMCn | <PORTn_base> + $0400_H$ + n × 4 |
| JTAG | | JPMC0 | <JPORT0_base> + $0040_H$ |
| PORT | Port mode control set/reset register | PMCSRn | <PORTn_base> + $0900_H$ + n × 4 |
| JTAG | | JPMCSR0 | <JPORT0_base> + $0090_H$ |
| PORT | Port IP control register | PIPCn | <PORTn_base> + $4200_H$ + n × 4 |
| PORT | Port mode register | PMn | <PORTn_base> + $0300_H$ + n × 4 |
| | | APMn | <PORTn_base> + $03C8_H$ + n × 4 |
| JTAG | | JPM0 | <JPORT0_base> + $0030_H$ |
| PORT | Port mode set/reset register | PMSRn | <PORTn_base> + $0800_H$ + n × 4 |
| | | APMSRn | <PORTn_base> + $08C8_H$ + n × 4 |
| JTAG | | JPMSR0 | <JPORT0_base> + $0080_H$ |
| PORT | Port input buffer control register | PIBCn | <PORTn_base> + $4000_H$ + n × 4 |
| | | APIBCn | <PORTn_base> + $40C8_H$ + n × 4 |
| JTAG | | JPIBC0 | <JPORT0_base> + $0400_H$ |
| PORT | | IPIBC0 | <PORTn_base> + $40F0_H$ |
| PORT | Port function control register | PFCn | <PORTn_base> + $0500_H$ + n × 4 |
| JTAG | | JPFC0 | <JPORT0_base> + $0050_H$ |
| PORT | Port function control expansion register | PFCEn | <PORTn_base> + $0600_H$ + n × 4 |
| JTAG | | JPFCE0 | <JPORT0_base> + $0060_H$ |
| PORT | Port function control additional expansion register | PFCAEn | <PORTn_base> + $0A00_H$ + n × 4 |
| **Pin data input/output** | | | |
| PORT | Port bidirection control register | PBDCn | <PORTn_base> + $4100_H$ + n × 4 |
| | | APBDCn | <PORTn_base> + $41C8_H$ + n × 4 |
| JTAG | | JPBDC0 | <JPORT0_base> + $0410_H$ |
| PORT | Port pin read register | PPRn | <PORTn_base> + $0200_H$ + n × 4 |
| | | APPRn | <PORTn_base> + $02C8_H$ + n × 4 |
| JTAG | | JPPR0 | <JPORT0_base> + $0020_H$ |
| PORT | | IPPR0 | <PORTn_base> + $02F0_H$ |

Table 2.13 Port Group Configuration Registers (2/2)

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| PORT | Port register | Pn | <PORTn_base> + $0000_H$ + n × 4 |
| | | APn | <PORTn_base> + $00C8_H$ + n × 4 |
| JTAG | | JP0 | <JPORT0_base> + $0000_H$ |
| PORT | Port NOT register | PNOTn | <PORTn_base> + $0700_H$ + n × 4 |
| | | APNOTn | <PORTn_base> + $07C8_H$ + n × 4 |
| JTAG | | JPNOT0 | <JPORT0_base> + $0070_H$ |
| PORT | Port set/reset register | PSRn | <PORTn_base> + $0100_H$ + n × 4 |
| | | APSRn | <PORTn_base> + $01C8_H$ + n × 4 |
| JTAG | | JPSR0 | <JPORT0_base> + $0010_H$ |
| **Configuration of electrical characteristics** | | | |
| PORT | Pull-up option register | PUn | <PORTn_base> + $4300_H$ + n × 4 |
| JTAG | | JPU0 | <JPORT0_base> + $0430_H$ |
| PORT | Pull-down option register | PDn | <PORTn_base> + $4400_H$ + n × 4 |
| JTAG | | JPD0 | <JPORT0_base> + $0440_H$ |
| PORT | Port drive strength control register | PDSCn | <PORTn_base> + $4600_H$ + n × 4 |
| JTAG | | JPDSC0 | <JPORT0_base> + $0460_H$ |
| PORT | Port open drain control register | PODCn | <PORTn_base> + $4500_H$ + n × 4 |
| JTAG | | JPODC0 | <JPORT0_base> + $0450_H$ |
| PORT | Port input buffer selection register | PISn | <PORTn_base> + $4700_H$ + n × 4 |
| JTAG | | JPIS0 | <JPORT0_base> + $0470_H$ |
| JTAG | Port input buffer selection advanced register | JPISA0 | <JPORT0_base> + $04A0_H$ |
| **Port protection** | | | |
| PORT | Port protection command register | PPCMDn | <PORTn_base> + $4C00_H$ + n × 4 |
| JTAG | | JPPCMD0 | <JPORT0_base> + $04C0_H$ |
| PORT | Port protection status register | PPROTSn | <PORTn_base> + $4B00_H$ + n × 4 |
| JTAG | | JPPROTS0 | <JPORT0_base> + $04B0_H$ |

### Index n

In **Table 2.13, Port Group Configuration Registers**, the index "n" in register symbols denotes the actual indices of the individual port groups. For example, PMCn generically indicates a port mode control register for port group n (Pn). The values for n differ according to the number of pins on the device in the way shown in **Table 2.14**.

Table 2.14 Number of Pins on the Device, Name of Port Groups, and Values for "n" in Register Symbols

| Number of Pins on the Device | Port Groups | Values for "n" |
|---|---|---|
| 100 pins | P0, P8, P9, P10, P11 | 0, 8, 9, 10, 11 |
| | AP0 | 0 |
| 144 pins | P0, P1, P8, P9, P10, P11, P12, P18, P20 | 0, 1, 8, 9, 10, 11, 12, 18, 20 |
| | AP0, AP1 | 0, 1 |
| 176 pins | P0, P1, P2, P8, P9, P10, P11, P12, P18, P20 | 0, 1, 2, 8, 9, 10, 11, 12, 18, 20 |
| | AP0, AP1 | 0, 1 |

### JTAG port registers

JTAG port registers are not explicitly described in the following register descriptions.

All descriptions (except for those of the PFCAEn register and PIPCn register) apply to JTAG port registers. Note, however, that the JTAG port register base address differs from that of regular ports.

### Value after reset

The values after reset depend on the ports. For the values after reset, see the register descriptions in the following pages.

## 2.9.2 Pin Function Configuration

### 2.9.2.1 PMCn / JPMC0 — Port Mode Control Register

This register specifies whether the individual pins of port group n are in port mode or in alternative mode.

**Access:** PMCn: This register can be read or written in 16-bit units.
JPMC0: This register can be read or written in 8-bit units.

**Address:** PMCn: <PORTn_base> + $0400_H$ + n × 4 (n = 0, 1, 2, 8, 9, 10, 11, 12, 18, 20)
JPMC0: <JPORT0_base> + $0040_H$*1

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMCn_15 | PMCn_14 | PMCn_13 | PMCn_12 | PMCn_11 | PMCn_10 | PMCn_9 | PMCn_8 | PMCn_7 | PMCn_6 | PMCn_5 | PMCn_4 | PMCn_3 | PMCn_2 | PMCn_1 | PMCn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.47, Control Registers (P8)**, **Table 2.49, Control Registers (P9)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, **Table 2.57, Control Registers (P18)**, and **Table 2.59, Control Registers (P20)**.

**Table 2.15 PMCn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | PMCn_[15:0] | Specifies the operation mode of the corresponding pin.<br>0: Port mode<br>1: Alternative mode |

**CAUTIONS**

1. I/O is not controlled by only setting alternative mode (PMCn.PMCn_m bit = 1). If the alternative function requires direct I/O control, also set the PIPCn.PIPCn_m bit to 1.
2. If a port is to be used as an input pin in alternative mode, the signals from some pins will pass through a noise filter. These pins may require the setting of the FCLA0CTLm_<name>, DNFA<name>CTL and the DNFA<name>EN register. For details, see **Section 2.12, Noise Filter & Edge/Level Detector**, and **Section 2.13, Description of Port Noise Filter & Edge/Level Detection**.

**NOTE**

The control bits of the JTAG port mode control register (JPMC0) are JPMC0_[7:0].

### 2.9.2.2 PMCSRn / JPMCSR0 — Port Mode Control Set/Reset Register

This register provides an alternative method to write data to the PMCn register.

The upper 16 bits of PMCSRn act as a mask which specifies whether or not the value of PMCn.PMCn_m is set by the corresponding bit in the lower 16 bits of PMCSRn.

**Access:** PMCSRn: This register can be read or written in 32-bit units. Bits 31 to 16 are always read as $0000_H$. Reading bits 15 to 0 returns the value of register PMCn.
JPMCSR0: This register can be read or written in 32-bit units. Bits 31 to 8 are always read as $000000_H$. Reading bits 7 to 0 returns the value of register JPMC0.

**Address:** PMCSRn: <PORTn_base> + $0900_H$ + n × 4 (n = 0, 1, 2, 8, 9, 10, 11, 12, 18, 20)
JPMCSR0: <JPORT0_base> + $0090_H$*1

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMCSRn_31 | PMCSRn_30 | PMCSRn_29 | PMCSRn_28 | PMCSRn_27 | PMCSRn_26 | PMCSRn_25 | PMCSRn_24 | PMCSRn_23 | PMCSRn_22 | PMCSRn_21 | PMCSRn_20 | PMCSRn_19 | PMCSRn_18 | PMCSRn_17 | PMCSRn_16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMCSRn_15 | PMCSRn_14 | PMCSRn_13 | PMCSRn_12 | PMCSRn_11 | PMCSRn_10 | PMCSRn_9 | PMCSRn_8 | PMCSRn_7 | PMCSRn_6 | PMCSRn_5 | PMCSRn_4 | PMCSRn_3 | PMCSRn_2 | PMCSRn_1 | PMCSRn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.47, Control Registers (P8)**, **Table 2.49, Control Registers (P9)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, **Table 2.57, Control Registers (P18)**, and **Table 2.59, Control Registers (P20)**.

**Table 2.16 PMCSRn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | PMCSRn_ [31:16] | Enable bits that specify whether the value of the corresponding lower bit PMCSRn_m (PMCSRn_[15:0]) is written to PMCn_m.<br>0: PMCn_m is not affected by PMCSRn_m.<br>1: PMCn_m is PMCSRn_m.<br>Example:<br>If PMCSRn.PMCSRn_31 = 1, the value of bit PMCSRn.PMCSRn_15 is written to bit PMCn.PMCn_15. |
| 15 to 0 | PMCSRn_ [15:0] | Data bits that specify the value of PMCn_m if PMCSRn_m of the corresponding upper bit (PMCSRn_[31:16]) is 1.<br>0: PMCn_m is 0.<br>1: PMCn_m is 1. |

**NOTE**

The control bits of the JTAG port mode control set/reset register (JPMCSR0) are JPMCSR0_[31:0].

### 2.9.2.3 PIPCn — Port IP Control Register

This register specifies whether the I/O direction of the Pn_m pin is controlled by the port mode register PMn.PMn_m or by an alternative function.

If the Pn_m pin is operated in alternative mode (PMCn.PMCn_m = 1) and the alternative function requires direct control of the I/O direction, then PIPCn.PIPCn_m must be set to 1 as well. This transfers I/O control to the alternative function and overrules the PMn.PMn_m setting.

Regarding the alternative functions for which the PIPC register must be set, see **Section 2.11, Port (Special I/O) Function Overview**

**Access:** This register can be read or written in 16-bit units.

**Address:** PIPCn: <PORTn_base> + $4200_H$ + n × 4 (n = 0, 10, 11) *1

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PIPCn_15 | PIPCn_14 | PIPCn_13 | PIPCn_12 | PIPCn_11 | PIPCn_10 | PIPCn_9 | PIPCn_8 | PIPCn_7 | PIPCn_6 | PIPCn_5 | PIPCn_4 | PIPCn_3 | PIPCn_2 | PIPCn_1 | PIPCn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.41, Control Registers (P0)**, **Table 2.51, Control Registers (P10)**, and **Table 2.53, Control Registers (P11)**.

**Table 2.17 PIPCn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | PIPCn_[15:0] | Specifies the I/O mode.<br>0: I/O mode is selected by PMn.PMn_m (software I/O control).<br>1: I/O mode is selected by the peripheral function (direct I/O control). |

### 2.9.2.4 PMn / APMn / JPM0 — Port Mode Register

This register specifies whether the individual pins of the port group n are in input mode or in output mode.

**Access:** PMn, APMn: These registers can be read or written in 16-bit units.
JPM0: This register can be read or written in 8-bit units.

**Address:** PMn: <PORTn_base> + $0300_H$ + n × 4 (n = 0, 1, 2, 8, 9, 10, 11, 12, 18, 20)
APMn: <PORTn_base> + $03C8_H$ + n × 4 (n = 0, 1)
JPM0: <JPORT0_base> + $0030_H$*1

**Value after reset:** $FFFF_H$*2

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMn_15 | PMn_14 | PMn_13 | PMn_12 | PMn_11 | PMn_10 | PMn_9 | PMn_8 | PMn_7 | PMn_6 | PMn_5 | PMn_4 | PMn_3 | PMn_2 | PMn_1 | PMn_0 |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1*3 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.47, Control Registers (P8)**, **Table 2.49, Control Registers (P9)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, **Table 2.57, Control Registers (P18)**, **Table 2.59, Control Registers (P20)**, **Table 2.61, Control Registers (AP0)**, and **Table 2.63, Control Registers (AP1)**.

Note 2. The PM8 register is as follows.
When the OPBT0.RESETOUTEN = 1, the PM8 register is $FFBF_H$.
When the OPBT0.RESETOUTEN = 0, the PM8 register is $FFFF_H$.

Note 3. The PM8_6 bit is as follows.
When the OPBT0.RESETOUTEN = 1, the PM8_6 bit is 0.
When the OPBT0.RESETOUTEN = 0, the PM8_6 bit is 1.

**Table 2.18 PMn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | PMn_[15:0] | Specifies input/output mode of the corresponding pin.<br>0: Output mode (output enabled)<br>1: Input mode (output disabled) |

**NOTES**

1. To use a port in input port mode (PMCn.PMCn_m = 0 and PMn.PMn_m = 1), the input buffer must be enabled (PIBCn.PIBCn_m = 1).
2. By default, PMn.PMn_m specifies the I/O direction in port mode (PMCn.PMCn_m = 0) and alternative mode (PMCn.PMCn_m=1), since PIPCn.PIPCn_m = 0 (I/O mode is controlled by PMn.PMn_m) after reset.
3. The control bits of the analog port register (APMn) are APMn_[15:0].
4. The control bits of the JTAG port mode register (JPM0) are JPM0_[7:0].

### 2.9.2.5 PMSRn / APMSRn / JPMSR0 — Port Mode Set/Reset Register

This register provides an alternative method to write data to the PMn register.

The upper 16 bits of PMSRn act as a mask which specifies whether or not the value PMn.PMn_m is set by the corresponding bit in the lower 16 bits of PMSRn.

**Access:** PMSRn, APMSRn: These registers can be read or written in 32-bit units. Bits 31 to 16 are always read as $0000_H$.
Reading bits 15 to 0 returns the value of registers PMn and APMn.
JPMSR0: This registers can be read or written in 32-bit units. Bits 31 to 8 are always read as $000000_H$.
Reading bits 7 to 0 returns the value of register JPM0.

**Address:** PMSRn: <PORTn_base> + $0800_H$ + n × 4 (n = 0, 1, 2, 8, 9, 10, 11, 12, 18, 20)
APMSRn: <PORTn_base> + $08C8_H$ + n × 4 (n = 0, 1)
JPMSR0: <JPORT0_base> + $0080_H$*1

**Value after reset:** 0000 $FFFF_H$*2

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMSRn_31 | PMSRn_30 | PMSRn_29 | PMSRn_28 | PMSRn_27 | PMSRn_26 | PMSRn_25 | PMSRn_24 | PMSRn_23 | PMSRn_22 | PMSRn_21 | PMSRn_20 | PMSRn_19 | PMSRn_18 | PMSRn_17 | PMSRn_16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMSRn_15 | PMSRn_14 | PMSRn_13 | PMSRn_12 | PMSRn_11 | PMSRn_10 | PMSRn_9 | PMSRn_8 | PMSRn_7 | PMSRn_6 | PMSRn_5 | PMSRn_4 | PMSRn_3 | PMSRn_2 | PMSRn_1 | PMSRn_0 |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1*3 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.47, Control Registers (P8)**, **Table 2.49, Control Registers (P9)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, **Table 2.57, Control Registers (P18)**, **Table 2.59, Control Registers (P20)**, **Table 2.61, Control Registers (AP0)**, and **Table 2.63, Control Registers (AP1)**.

Note 2. The PMSR8 register is as follows.
When the OPBT0.RESETOUTEN = 1, the PMSR8 register is 0000 $FFBF_H$.
When the OPBT0.RESETOUTEN = 0, the PMSR8 register is 0000 $FFFF_H$.

Note 3. The PMSR8_6 bit is as follows.
When the OPBT0.RESETOUTEN = 1, the PMSR8_6 bit is 0.
When the OPBT0.RESETOUTEN = 0, the PMSR8_6 bit is 1.

**Table 2.19 PMSRn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | PMSRn_[31:16] | Enable bits that specify whether the value of the corresponding lower bit PMSRn_m (PMSRn_[15:0]) is written to PMn_m.<br>0: PMn_m is not affected by PMSRn_m.<br>1: PMn_m is PMSRn_m.<br>Example:<br>If PMSRn.PMSRn_31 = 1, the value of bit PMSRn.PMSRn_15 is written to bit PMn.PMn_15. |
| 15 to 0 | PMSRn_[15:0] | Data bits that specify the value of PMn_m if PMSRn_m of the corresponding upper bit (PMSRn_[31:16]) is 1.<br>0: PMn_m is 0.<br>1: PMn_m is 1. |

**NOTES**

1. The control bits of the JTAG port mode set/reset register (JPMSR0) are JPMSR0_[31:0].
2. The control bits of the analog port mode set/reset register (APMSRn) are APMSRn_[31:0].

### 2.9.2.6 PIBCn / APIBCn / JPIBC0 / IPIBC0 — Port Input Buffer Control Register

In input port mode (PMCn.PMCn_m = 0 and PMn.PMn_m = 1), this register enables the port pin's input buffer.

**Access:** PIBCn, APIBCn, IPIBC0: These registers can be read or written in 16-bit units.
JPIBC0: This register can be read or written in 8-bit units.

**Address:** PIBCn: <PORTn_base> + $4000_H$ + n × 4 (n = 0, 1, 2, 8, 9, 10, 11, 12, 18, 20)
APIBCn: <PORTn_base> + $40C8_H$+ n × 4 (n = 0, 1)
JPIBC0: <JPORT0_base> + $0400_H$
IPIBC0: <PORTn_base> + $40F0_H$*1

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PIBCn_15 | PIBCn_14 | PIBCn_13 | PIBCn_12 | PIBCn_11 | PIBCn_10 | PIBCn_9 | PIBCn_8 | PIBCn_7 | PIBCn_6 | PIBCn_5 | PIBCn_4 | PIBCn_3 | PIBCn_2 | PIBCn_1 | PIBCn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.47, Control Registers (P8)**, **Table 2.49, Control Registers (P9)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, **Table 2.57, Control Registers (P18)**, **Table 2.59, Control Registers (P20)**, **Table 2.61, Control Registers (AP0)**, **Table 2.63, Control Registers (AP1)**, and **Table 2.65, Control Registers (IP0)**.

**Table 2.20 PIBCn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | PIBCn_[15:0] | Enables/disables the input buffer.<br>0: Input buffer disabled<br>1: Input buffer enabled |

**NOTES**

1. When the input buffer is disabled, through current does not flow even when the pin level is Hi-Z. Thus the pin does not need to be fixed to a high or low level externally.
2. The control bits of the JTAG port input buffer control register (JPIBC0) are JPIBC0_[7:0].

**CAUTION**

Settings in this register are overruled in bidirectional mode (PBDCn.PBDCn_m = 1).

### 2.9.2.7 PFCn / JPFC0 — Port Function Control Register

This register, together with register PFCEn and PFCAEn, specifies an alternative function of the pins.

Some alternative functions directly control the I/O of the Pn_m pin. For such alternative functions, PIPCn.PIPCn_m must be set to 1 and the I/O is selected by the peripheral function.

For other alternative functions, input/output must be specified by PMn.PMn_m.

**Access:** PFCn: This register can be read or written in 16-bit units.
JPFC0: This register can be read or written in 8-bit units.

**Address:** PFCn: <PORTn_base> + $0500_H$ + n × 4 (n = 0, 1, 2, 8, 9, 10, 11, 12, 18, 20)
JPFC0: <JPORT0_base> + $0050_H$*1

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PFCn_15 | PFCn_14 | PFCn_13 | PFCn_12 | PFCn_11 | PFCn_10 | PFCn_9 | PFCn_8 | PFCn_7 | PFCn_6 | PFCn_5 | PFCn_4 | PFCn_3 | PFCn_2 | PFCn_1 | PFCn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.47, Control Registers (P8)**, **Table 2.49, Control Registers (P9)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, **Table 2.57, Control Registers (P18)**, and **Table 2.59, Control Registers (P20)**..

**Table 2.21 PFCn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | PFCn_[15:0] | Specifies the alternative function of the pin.<br>For details, see **Table 2.24, Setting Alternative Functions**. |

**NOTE**

The control bits of the JTAG port function control register (JPFC0) are JPFC0_[7:0].

### 2.9.2.8 PFCEn / JPFCE0 — Port Function Control Expansion Register

This register, together with register PFCn and PFCAEn, specifies an alternative function of the pins.

Some alternative functions directly control the I/O of the Pn_m pin. For such alternative functions, PIPCn.PIPCn_m must be set to 1 and the I/O is specified by the peripheral function.

For other alternative functions, input/output must be specified by PMn.PMn_m.

**Access:** PFCEn: This register can be read or written in 16-bit units.
JPFCE0: This register can be read or written in 8-bit units.

**Address:** PFCEn: <PORTn_base> + $0600_H$ + n × 4 (n = 0, 1, 2, 8, 9, 10, 11, 12, 20)
JPFCE0: <JPORT0_base> + $0060_H$*1

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PFCEn_15 | PFCEn_14 | PFCEn_13 | PFCEn_12 | PFCEn_11 | PFCEn_10 | PFCEn_9 | PFCEn_8 | PFCEn_7 | PFCEn_6 | PFCEn_5 | PFCEn_4 | PFCEn_3 | PFCEn_2 | PFCEn_1 | PFCEn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**:**Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**,**Table 2.47, Control Registers (P8)**, **Table 2.49, Control Registers (P9)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, and **Table 2.59, Control Registers (P20)**.

**Table 2.22 PFCEn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | PFCEn_[15:0] | Specifies the alternative function of the pin.<br>For details, see **Table 2.24, Setting Alternative Functions**. |

**NOTE**

The control bits of the JTAG port function control register (JPFCE0) are JPFCE0_[7:0].

### 2.9.2.9 PFCAEn — Port Function Control Additional Expansion Register

This register selects the alternative peripheral functions together with PFCEn, PFCn registers.

Some alternative functions directly control the I/O of the Pn_m pin. For such alternative functions, PIPCn.PIPCn_m must be set to 1 and the I/O is specified by the peripheral function.

For other alternative functions, input/output must be specified by PMn.PMn_m.

**Access:** This register can be read or written in 16-bit units.

**Address:** PFCAEn: <PORTn_base> + $0A00_H$ + n × 4 (n = 0, 1, 2, 10, 11, 12, 20)*1

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PFCAEn_15 | PFCAEn_14 | PFCAEn_13 | PFCAEn_12 | PFCAEn_11 | PFCAEn_10 | PFCAEn_9 | PFCAEn_8 | PFCAEn_7 | PFCAEn_6 | PFCAEn_5 | PFCAEn_4 | PFCAEn_3 | PFCAEn_2 | PFCAEn_1 | PFCAEn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, and **Table 2.59, Control Registers (P20)**.

**Table 2.23 PFCAEn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | PFCAEn_[15:0] | Specifies the alternative function of the pin.<br>For details, see **Table 2.24, Setting Alternative Functions**. |

**Table 2.24 Setting Alternative Functions**

| PFCAEn_m | PFCEn_m | PFCn_m | PMn_m | Function |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | Alternative input mode 1 |
| | | | 0 | Alternative output mode 1 |
| | | 1 | 1 | Alternative input mode 2 |
| | | | 0 | Alternative output mode 2 |
| | 1 | 0 | 1 | Alternative input mode 3 |
| | | | 0 | Alternative output mode 3 |
| | | 1 | 1 | Alternative input mode 4 |
| | | | 0 | Alternative output mode 4 |
| 1 | 0 | 0 | 1 | Alternative input mode 5 |
| | | | 0 | Alternative output mode 5 |
| | | 1 | 1 | Alternative input mode 6 |
| | | | 0 | Alternative output mode 6 |
| | 1 | 0 | 1 | Alternative input mode 7 |
| | | | 0 | Alternative output mode 7 |
| | | 1 | X | Setting prohibited |

**CAUTION**

- After selecting the alternative function by the PFCn_m, PFCEn_m, or PFCAEn_m bit, set the PMCn_m bit to “1”.
- With this product, the I/O of some functions is assigned to two or more pins, but a specific pin function can only be set to one pin at a time. Setting the same pin function to two or more pins at the same time is prohibited.
  For example, if the a/b/c pin is used as b, the b/d/e pin cannot be used as b. In this case, the b/d/e pin must be configured as a pin function other than b.

**NOTE**

For more details on the assignment of each function, see **Sections 2.10.1** to **2.10.14**.

## 2.9.3 Pin Data Input/Output

### 2.9.3.1 PBDCn / APBDCn / JPBDC0 — Port Bidirection Control Register

This register enables the input buffer in output mode and sets the port to bidirectional mode. In bidirectional mode, the level of the signal on a Pn_m pin can be read from PPRn.PPRn_m.

**Access:** PBDCn, APBDCn: These registers can be read or written in 16-bit units.
JPBDC0: This register can be read or written in 8-bit units.

**Address:** PBDCn: <PORTn_base> + $4100_H$ + n × 4 (n = 0, 1, 2, 8, 9, 10, 11, 12, 18, 20)
APBDCn: <PORTn_base> + $41C8_H$ + n × 4 (n = 0, 1)
JPBDC0: <JPORT0_base> + $0410_H$*1

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PBDCn_15 | PBDCn_14 | PBDCn_13 | PBDCn_12 | PBDCn_11 | PBDCn_10 | PBDCn_9 | PBDCn_8 | PBDCn_7 | PBDCn_6 | PBDCn_5 | PBDCn_4 | PBDCn_3 | PBDCn_2 | PBDCn_1 | PBDCn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.47, Control Registers (P8)**, **Table 2.49, Control Registers (P9)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, **Table 2.57, Control Registers (P18)**, **Table 2.59, Control Registers (P20)**, **Table 2.61, Control Registers (AP0)**, and **Table 2.63, Control Registers (AP1)**.

**Table 2.25 PBDCn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | PBDCn[15:0] | Enables/disables bidirectional mode of the corresponding pin.<br>0: Bidirectional mode disabled<br>1: Bidirectional mode enabled |

**CAUTION**

When the Pn_m port is used for the alternative output function (PMCn.PMCn_m = 1, PMn.PMn_m = 0), the level of the Pn_m pin can be read from PPRn.PPRn_m by enabling the bidirectional mode (PBDCn.PBDCn_m = 1).

However, output of that alternative output function is input to the alternative input function of the same pin (the alternative input function set by PFCn.PFCn_m, PFCEn.PFCEn_m, and PFCAEn.PFCAEn_m). If the alternative input function in question is being used by another pin, the alternative input function is not guaranteed.

**NOTE**

The control bits of the JTAG port bidirection control register (JPBDC0) are JPBDC0_[7:0].

### 2.9.3.2 PPRn / APPRn / JPPR0 / IPPR0 — Port Pin Read Register

This register reflects the actual level of the Pn_m pin, whether it is the value of the Pn.Pn_m bit or the level of an alternative output function.

**Access:** PPRn, APPRn, IPPR0: These registers are read-only registers that can be read in 16-bit units.
JPPR0: This register is a read-only register that can be read in 8-bit units.

**Address:** PPRn: <PORTn_base> + $0200_H$ + n × 4 (n = 0, 1, 2, 8, 9, 10, 11, 12, 18, 20)
APPRn: <PORTn_base> + $02C8_H$ + n × 4 (n = 0, 1)
JPPR0: <JPORT0_base> + $0020_H$
IPPR0: <PORTn_base> + $02F0_H$*1

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PPRn_15 | PPRn_14 | PPRn_13 | PPRn_12 | PPRn_11 | PPRn_10 | PPRn_9 | PPRn_8 | PPRn_7 | PPRn_6 | PPRn_5 | PPRn_4 | PPRn_3 | PPRn_2 | PPRn_1 | PPRn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.47, Control Registers (P8)**, **Table 2.49, Control Registers (P9)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, **Table 2.57, Control Registers (P18)**, **Table 2.59, Control Registers (P20)**, **Table 2.61, Control Registers (AP0)**, **Table 2.63, Control Registers (AP1)**, and **Table 2.65, Control Registers (IP0)**.

**Table 2.26 PPRn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | PPRn_[15:0] | The Pn_m Pin, Pn.Pn_m value or alternative function output. |

**NOTES**

1. For the read values of the PPRn register, see **Section 2.7.4, Pin Data Input/Output**.
2. The control bits of the JTAG port pin read register (JPPR0) are JPPR0_[7:0].

### 2.9.3.3 Pn / APn / JP0 — Port Register

This register holds the Pn.Pn_m data to be output via the related Pn_m port in output port mode (PMCn.PMCn_m = 0 and PMn.PMn_m = 0).

**Access:** Pn, APn: These registers can be read or written in 16-bit units.
JP0: This register can be read or written in 8-bit units.

**Address:** Pn: <PORTn_base> + $0000_H$ + n × 4 (n = 0, 1, 2, 8, 9, 10, 11, 12, 18, 20)
APn: <PORTn_base> + $00C8_H$ + n × 4 (n = 0, 1)
JP0: <JPORT0_base> + $0000_H$*1

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Pn_15 | Pn_14 | Pn_13 | Pn_12 | Pn_11 | Pn_10 | Pn_9 | Pn_8 | Pn_7 | Pn_6 | Pn_5 | Pn_4 | Pn_3 | Pn_2 | Pn_1 | Pn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.47, Control Registers (P8)**, **Table 2.49, Control Registers (P9)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, **Table 2.57, Control Registers (P18)**, **Table 2.59, Control Registers (P20)**, **Table 2.61, Control Registers (AP0)**, and **Table 2.63, Control Registers (AP1)**.

**Table 2.27 Pn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | Pn_[15:0] | Sets the output level of the Pn_m pin (m = 0 to 15).<br>0: Outputs low level<br>1: Outputs high level |

**NOTE**

The control bits of the JTAG port register (JP0) are JP0_[7:0].

### 2.9.3.4 PNOTn / APNOTn / JPNOT0 — Port NOT Register

This register allows the Pn_m bit of the port register Pn to be inverted without directly writing to Pn.

**Access:** PNOTn, APNOTn: These registers are write-only registers that can be written in 16-bit units. When read, $0000_H$ is returned.
JPNOT0: This register is a write-only register that can be written in 8-bit units. When read, $00_H$ is returned.

**Address:** PNOTn: <PORTn_base> + $0700_H$ + n × 4 (n = 0, 1, 2, 8, 9, 10, 11, 12, 18, 20)
APNOTn: <PORTn_base> + $07C8_H$ + n × 4 (n = 0, 1)
JPNOT0: <JPORT0_base> + $0070_H$*1

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PNOTn_15 | PNOTn_14 | PNOTn_13 | PNOTn_12 | PNOTn_11 | PNOTn_10 | PNOTn_9 | PNOTn_8 | PNOTn_7 | PNOTn_6 | PNOTn_5 | PNOTn_4 | PNOTn_3 | PNOTn_2 | PNOTn_1 | PNOTn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.47, Control Registers (P8)**, **Table 2.49, Control Registers (P9)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, **Table 2.57, Control Registers (P18)**, **Table 2.59, Control Registers (P20)**, **Table 2.61, Control Registers (AP0)**, and **Table 2.63, Control Registers (AP1)**.

**Table 2.28 PNOTn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | PNOTn_[15:0] | Specifies if Pn.Pn_m is inverted.<br>0: Pn.Pn_m is not inverted (Pn_m → Pn_m)<br>1: Pn.Pn_m is inverted ($\overline{Pn_m}$ → Pn_m) |

**NOTE**

The control bits of the JTAG port NOT register are JPNOT0_[7:0].

### 2.9.3.5 PSRn / APSRn / JPSR0 — Port Set/Reset Register

This register provides an alternative method to write data to the Pn register.

The upper 16 bits of PSRn act as a mask which specifies whether or not the value Pn.Pn_m is set by the corresponding bit in the lower 16 bits of PSRn.

**Access:** PSRn, APSRn: These registers can be read or written in 32-bit units. Bits 31 to 16 are always read as $0000_H$. Reading bits 15 to 0 returns the value of registers Pn and APn.
JPSR0: This register can be read or written in 32-bit units. Bits 31 to 8 are always read as $000000_H$. Reading bits 7 to 0 returns the value of register JP0.

**Address:** PSRn: <PORTn_base> + $0100_H$ + n × 4 (n = 0, 1, 2, 8, 9, 10, 11, 12, 18, 20)
APSRn: <PORTn_base> + $01C8_H$ + n × 4 (n = 0, 1)
JPSR0: <JPORT0_base> + $0010_H$*1

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSRn_31 | PSRn_30 | PSRn_29 | PSRn_28 | PSRn_27 | PSRn_26 | PSRn_25 | PSRn_24 | PSRn_23 | PSRn_22 | PSRn_21 | PSRn_20 | PSRn_19 | PSRn_18 | PSRn_17 | PSRn_16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSRn_15 | PSRn_14 | PSRn_13 | PSRn_12 | PSRn_11 | PSRn_10 | PSRn_9 | PSRn_8 | PSRn_7 | PSRn_6 | PSRn_5 | PSRn_4 | PSRn_3 | PSRn_2 | PSRn_1 | PSRn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.47, Control Registers (P8)**, **Table 2.49, Control Registers (P9)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, **Table 2.57, Control Registers (P18)**, **Table 2.59, Control Registers (P20)**, **Table 2.61, Control Registers (AP0)**, and **Table 2.63, Control Registers (AP1)**.

**Table 2.29 PSRn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | PSRn_[31:16] | Specifies whether the value of the corresponding lower bit PSRn_m (PSRn_[15:0]) is written to Pn_m.<br>0: Pn_m is not affected by PSRn_m.<br>1: Pn_m is PSRn_m<br>Example:<br>If PSRn.PSRn_31 = 1, the value of bit PSRn.PSRn_15 is written to bit Pn.Pn_15. |
| 15 to 0 | PSRn_[15:0] | Specifies the Pn_m value if the corresponding upper bit (PSRn_[31:16]) PSRn_m is 1.<br>0: Pn_m = 0<br>1: Pn_m = 1 |

**NOTE**

The control bits of the JTAG port set/reset register (JPSR0) are JPSR0_[31:0].

## 2.9.4 Configuration of Electrical Characteristics

### 2.9.4.1 PUn / JPU0 — Pull-Up Option Register

This register specifies whether an internal pull-up resistor is connected to an input pin.

**Access:** PUn: This register can be read or written in 16-bit units.
JPU0: This register can be read or written in 8-bit units.

**Address:** PUn: <PORTn_base> + $4300_H$ + n × 4 (n = 0, 1, 2, 8, 9, 10, 11,12, 18, 20)
JPU0: <JPORT0_base> + $0430_H$*1

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PUn_15 | PUn_14 | PUn_13 | PUn_12 | PUn_11 | PUn_10 | PUn_9 | PUn_8 | PUn_7 | PUn_6 | PUn_5 | PUn_4 | PUn_3 | PUn_2 | PUn_1 | PUn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.47, Control Registers (P8)**, **Table 2.49, Control Registers (P9)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, **Table 2.57, Control Registers (P18)**, and **Table 2.59, Control Registers (P20)**.

**Table 2.30 PUn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | PUn_[15:0] | Specifies whether an internal pull-up resistor is connected to the corresponding pin.<br>0: No internal pull-up resistor connected<br>1: An internal pull-up resistor connected |

**NOTES**

1. If a pin is configured such that both an internal pull-up resistor (PUn.PUn_m = 1) and pull-down resistor (PDn.PDn_m = 1) are connected, the pull-down resistor is automatically selected and the pull-up resistor is not connected.
2. The pull-up resistor has no effect when the pin is operated in output mode.
3. The control bits of the JTAG pull-up option register (JPU0) are JPU0_[7:0].

### 2.9.4.2 PDn / JPD0 — Pull-Down Option Register

This register specifies whether to connect an internal pull-down resistor to an input pin.

**Access:** PDn: This register can be read or written in 16-bit units.
JPD0: This register can be read or written in 8-bit units.

**Address:** PDn: <PORTn_base> + $4400_H$ + n × 4 (n = 0, 1, 2, 8, 9, 10, 11, 12, 18, 20)
JPD0: <JPORT0_base> + $0440_H$*1

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PDn_15 | PDn_14 | PDn_13 | PDn_12 | PDn_11 | PDn_10 | PDn_9 | PDn_8 | PDn_7 | PDn_6 | PDn_5 | PDn_4 | PDn_3 | PDn_2 | PDn_1 | PDn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.47, Control Registers (P8)**, **Table 2.49, Control Registers (P9)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, **Table 2.57, Control Registers (P18)**, and **Table 2.59, Control Registers (P20)**.

**Table 2.31 PDn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | PDn_[15:0] | Specifies whether to connect an internal pull-down resistor to the corresponding pin:<br>0: No internal pull-down resistor connected<br>1: An internal pull-down resistor connected |

**NOTES**

1. If a pin is configured such that both an internal pull-up resistor (PUn.PUn_m = 1) and pull-down resistor (PDn.PDn_m = 1) are connected, the pull-down resistor is automatically selected and the pull-up resistor is not connected.
2. The internal pull-down resistor has no effect when the pin is operated in output mode.
3. The control bits of the JTAG pull-down option register (JPD0) are JPD0_[7:0].

### 2.9.4.3 PDSCn / JPDSC0 — Port Drive Strength Control Register

This register specifies the output driver strength of the port pin. This function selects the fast mode (high drive strength) or slow mode (low drive strength) of the output buffer. The correct write sequence using the PPCMDn register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**. Regarding the alternative functions for which the PDSC register needs to be set, see **Section 2.11.3.3, Output Buffer Control (PDSC)**.

**Access:** PDSCn, JPDSC0: These registers can be read or written in 32-bit units.

**Address:** PDSCn: <PORTn_base> + $4600_H$ + n × 4 (n = 0, 1, 2, 10, 11, 12, 18, 20)
JPDSC0: <JPORT0_base> + $0460_H$*1

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PDSCn_15 | PDSCn_14 | PDSCn_13 | PDSCn_12 | PDSCn_11 | PDSCn_10 | PDSCn_9 | PDSCn_8 | PDSCn_7 | PDSCn_6 | PDSCn_5 | PDSCn_4 | PDSCn_3 | PDSCn_2 | PDSCn_1 | PDSCn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, **Table 2.57, Control Registers (P18)**, and **Table 2.59, Control Registers (P20)**.

**Table 2.32 PDSCn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 15 to 0 | PDSCn_[15:0] | Specifies the port drive strength of the output buffer of the port pin.<br>0: Lower drive strength (when the frequency output from the pin is 10 MHz or below)<br>1: High drive strength (when the frequency output from the pin is 40 MHz or less). |

**NOTE**

The control bits of the JTAG port drive strength control register (JPDSC0) are JPDSC0_[31:0].

### 2.9.4.4 PODCn / JPODC0 — Port Open Drain Control Register

This register selects push-pull or open-drain as output buffer function. The correct write sequence using the PPCMDn and JPPCMD0 registers is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

**Access:** PODCn, JPODC0: These registers can be read or written in 32-bit units.

**Address:** PODCn: <PORTn_base> + $4500_H$ + n × 4 (n = 0, 1, 2, 8, 9, 10, 11, 12, 18, 20)
JPODC0: <JPORT0_base> + $0450_H$*1

**Value after reset:** 0000 $0000_H$*2

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PODCn_15 | PODCn_14 | PODCn_13 | PODCn_12 | PODCn_11 | PODCn_10 | PODCn_9 | PODCn_8 | PODCn_7 | PODCn_6 | PODCn_5 | PODCn_4 | PODCn_3 | PODCn_2 | PODCn_1 | PODCn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0*3 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.47, Control Registers (P8)**, **Table 2.49, Control Registers (P9)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, **Table 2.57, Control Registers (P18)**, and **Table 2.59, Control Registers (P20)**.

Note 2. The PODC8 register is as follows.
When the OPBT0.RESETOUTEN = 1, the PODC8 register is 0000 $0040_H$.
When the OPBT0.RESETOUTEN = 0, the PODC8 register is 0000 $0000_H$.

Note 3. The PODC8_6 bit is as follows.
When the OPBT0.RESETOUTEN = 1, the PODC8_6 bit is 1.
When the OPBT0.RESETOUTEN = 0, the PODC8_6 bit is 0.

**Table 2.33 PODCn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 15 to 0 | PODCn_[15:0] | Specifies the output buffer function.<br>0: Push-pull<br>1: Open-drain |

**NOTE**

The control bits of the JTAG port open drain control register (JPODC0) are JPODC0_[31:0].

### 2.9.4.5 PISn/JPIS0 — Port Input Buffer Selection Register

This register specifies the input buffer characteristics.

**Access:** PISn: This register can be read or written in 16-bit units.
JPIS0: This register can be read or written in 8-bit units.

**Address:** PISn: <PORTn_base> + $4700_H$ + n × 4 (n = 0, 1, 2, 8, 9, 10, 11, 12, 18, 20)
JPIS0: <JPORT0_base> + $0470_H$*1

**Value after reset:** $FFFF_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PISn_15 | PISn_14 | PISn_13 | PISn_12 | PISn_11 | PISn_10 | PISn_9 | PISn_8 | PISn_7 | PISn_6 | PISn_5 | PISn_4 | PISn_3 | PISn_2 | PISn_1 | PISn_0 |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**, **Table 2.41, Control Registers (P0)**, **Table 2.43, Control Registers (P1)**, **Table 2.45, Control Registers (P2)**, **Table 2.47, Control Registers (P8)**, **Table 2.49, Control Registers (P9)**, **Table 2.51, Control Registers (P10)**, **Table 2.53, Control Registers (P11)**, **Table 2.55, Control Registers (P12)**, **Table 2.57, Control Registers (P18)**, and **Table 2.59, Control Registers (P20)**.

**Table 2.34 PISn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | PISn_[15:0] | Specifies the input buffer Characteristics:<br>0: Type 1 (SHMT1)<br>1: Type 2 (SHMT4) |

**NOTE**

1. Details of the definition of type 1 and type 2 are given in **Section 2.11.3.2, Input Buffer Control (PISn/JPIS0, JPISA0)**. For details, also see the **Section 40, Electrical Characteristics** for input buffer characteristics.
2. The control bits of the JTAG port input buffer selection register (JPIS0) are JPIS0_[7:0].

### 2.9.4.6 JPISA0 — Port Input Buffer Selection Advanced Register

This register specifies the input buffer characteristics.

**Access:** This register can be read or written in 8-bit units.

**Address:** JPISA0: <JPORT0_base> + $04A0_H$*1

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | JPISA0_3 | JPISA0_2 | - | JPISA0_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R | R/W |

Note 1. The valid bit positions (value for the index m) vary depending on the number of pins for each device. See the following tables in **Section 2.10, Port (General I/O) Function Overview**: **Table 2.39, Control Registers (JP0)**.

**Table 2.35 JPISA0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4, 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3, 2, 0 | JPISA0_[3, 2, 0] | Specifies the input buffer characteristics:<br>0: Type 2 (SHMT4)<br>1: Type 5 (TTL) |

**Table 2.36 JTAG Port Input Selection Advanced Register Contents**

| JPISA0 | JPIS0 | Function |
|---|---|---|
| 0 | 0 | Type 1 input buffer is selected (SHMT1) |
| | 1 | Type 2 input buffer is selected (SHMT4) |
| 1 | X | Type 5 input buffer is selected (TTL) |

**NOTE**

Details of the definition of type 2 and type 5 are given in **Section 2.11.3.2, Input Buffer Control (PISn/JPIS0, JPISA0)**. For details, also see the **Section 40, Electrical Characteristics** for input buffer characteristics.

## 2.9.5 Port Register Protection

RH850/F1K has Port Protection Command Registers (PPCMDn) and Port Protection Status Registers (PPROTSn) which implement the Port Protection Cluster Function. For details on the registers, see **Section 5, Write-Protected Registers**.

## 2.9.6 Flowchart Examples for Port Settings

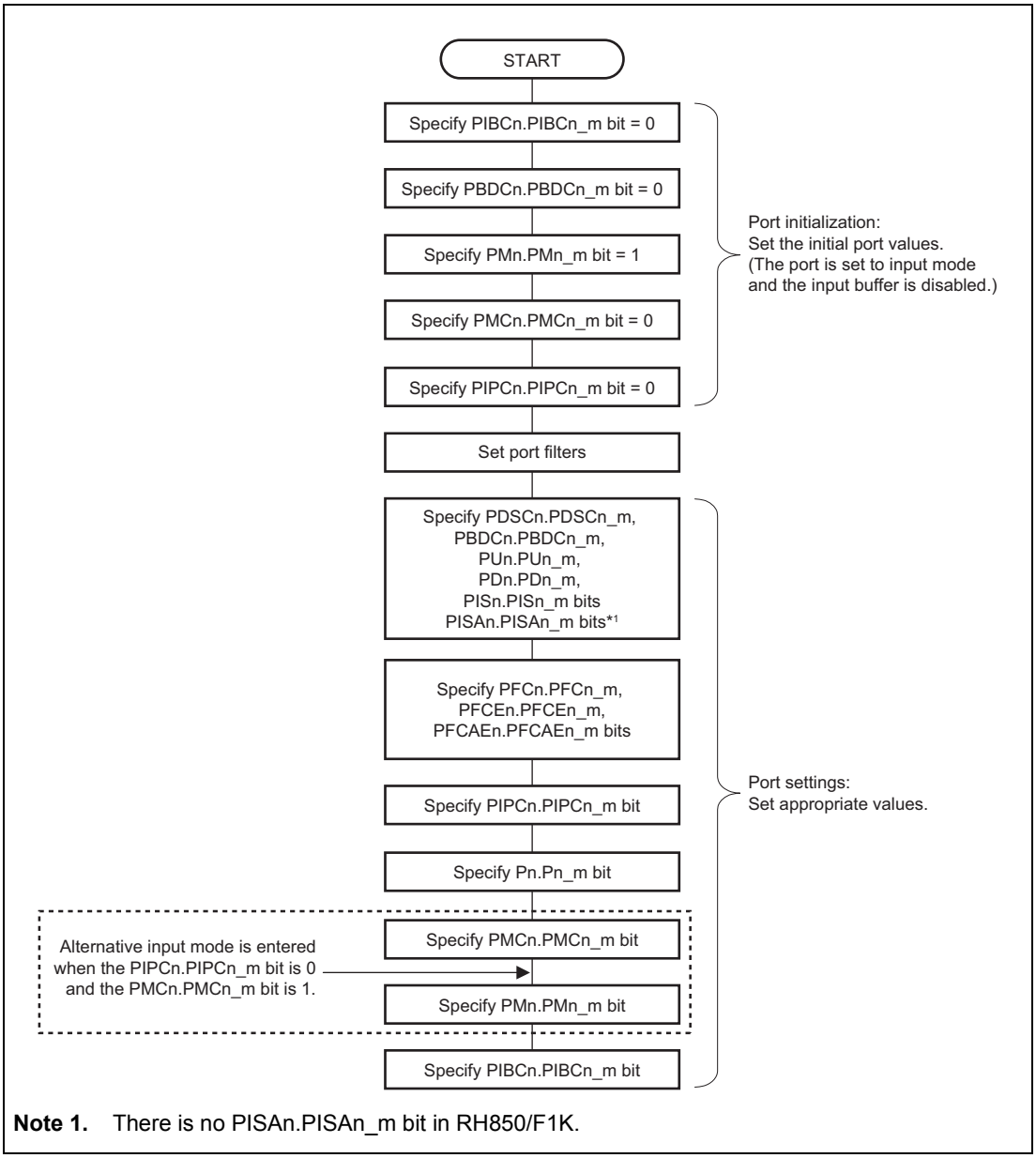
Examples of the port settings are shown in the flowchart below.

**CAUTION**

**If the port is set to the PIPCn.PIPCn_m bit = 0 and alternative output mode, the port might briefly enter alternative input mode. This will occur between when the PMCn.PMCn_m bit is set to 1 and when the PMn.PMn_m bit is set to 0. If an interrupt-related signal is specified as an alternate function of the port, the mode will temporarily become the alternative input mode, so either disable the interrupt in question, or specify that the interrupt is ignored.**

### 2.9.6.1 Batch Setting

An example of specifying batch port group settings is shown in the flowchart below.

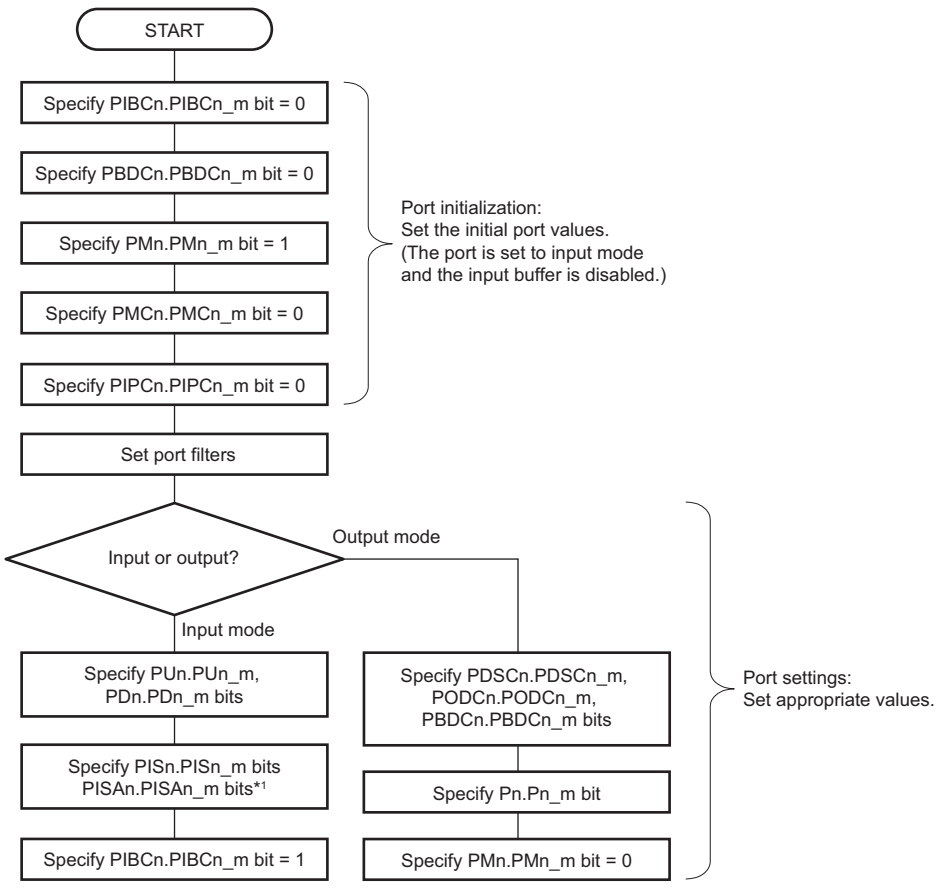



**Note 1.** There is no PISAn.PISAn_m bit in RH850/F1K.

**Figure 2.5 Example of Port Settings (When Specified in Batch)**

### 2.9.6.2 Individual Settings

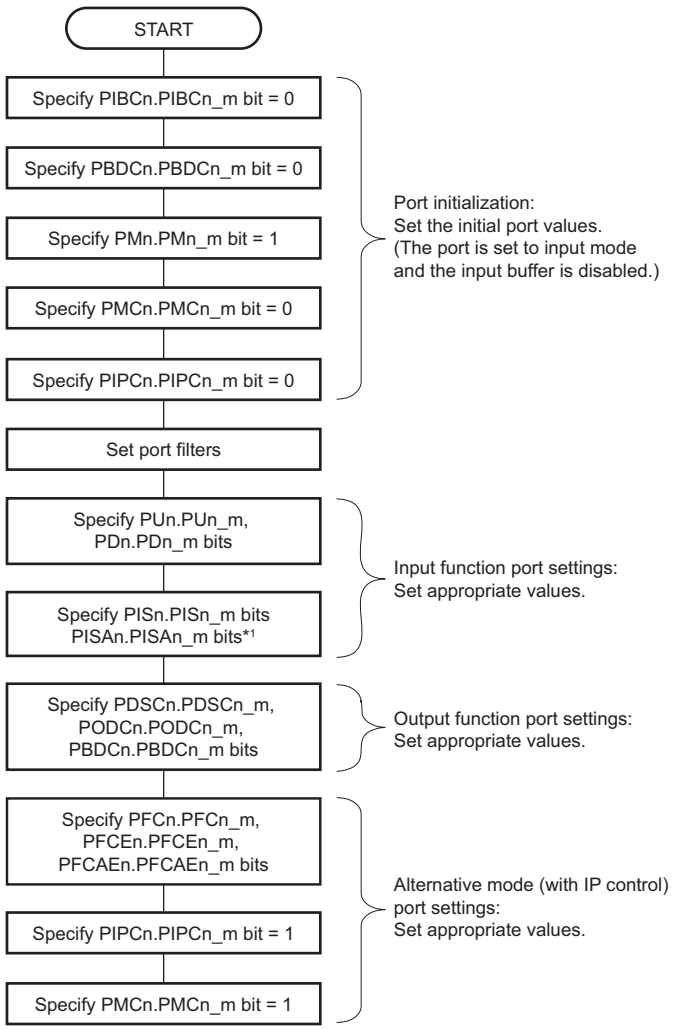
An example of specifying individual port settings is shown in the flowchart below.




**Note 1.** There is no PISAn.PISAn_m bit in RH850/F1K.

**Figure 2.6 Example of Port Settings (in Port Mode)**

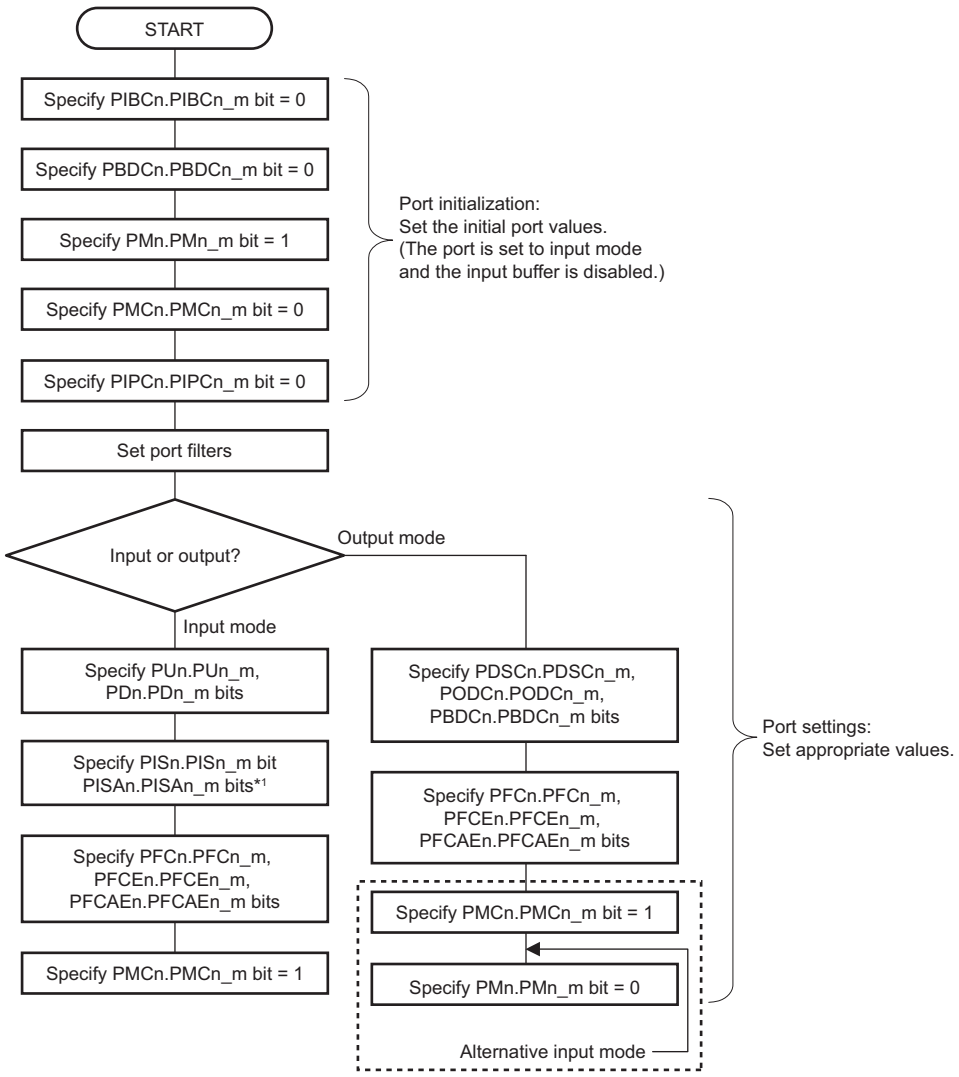
(a) With IP control




**Note 1.** There is no PISAn.PISAn_m bit in RH850/F1K.

**Figure 2.7 Example of Port Settings (in Alternative Mode) (1/2)**

(b) Without IP control




**Note 1.** There is no PISAn.PISAn_m bit in RH850/F1K.

**Figure 2.8 Example of Port Settings (in Alternative Mode) (2/2)**

## 2.10 Port (General I/O) Function Overview

This section explains the port (general I/O) functions and all the functions assigned to the ports. See the following pages for details.

In addition, whether the port mode is alternative mode or not can be selected by PMCn register setting. When PMCn.PMCn_m = 1, alternative functions are selected by the PFCn, PFCEn, and PFCAEn registers.

**Table 2.37 Port Function**

| Port | Pin Name | Size | Direction | Power Domain | Special Alternative Function | Device | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | 100 pins | 144 pins | 176 pins |
| JTAG Port 0 | JP0_0 - 5 | 6 bits | In/Out | AWO | JTAG, LPD | √ | — | — |
| | JP0_0 - 6 | 7 bits | | | | — | √ | √ |
| Port 0 | P0_0 - 14 | 15 bits | In/Out | AWO | | √ | √ | √ |
| Port 1 | P1_0 - 11 | 12 bits | In/Out | AWO | | — | √ | — |
| | P1_0 - 15 | 16 bits | | | | — | — | √ |
| Port 2 | P2_0 - 6 | 7 bits | In/Out | AWO | | — | — | √ |
| Port 8 | P8_0 - 12 | 13 bits | In/Out | AWO | ADCA0 (10-bit resolution)<br>RESETOUT | √ | √ | √ |
| Port 9 | P9_0 - 6 | 7 bits | In/Out | ISO | ADCA0 (10-bit resolution) | √ | √ | √ |
| Port 10 | P10_0 - 15 | 16 bits | In/Out | ISO | | √ | √ | √ |
| Port 11 | P11_0 - 7 | 8 bits | In/Out | ISO | | √ | — | — |
| | P11_0 - 15 | 16 bits | | | | — | √ | √ |
| Port 12 | P12_0 - 2 | 3 bits | In/Out | ISO | | — | √ | — |
| | P12_0 - 5 | 6 bits | | | | — | — | √ |
| Port 18 | P18_0 - 3 | 4 bits | In/Out | ISO | ADCA1 (10-bit resolution) | — | √ | — |
| | P18_0 - 7 | 8 bits | | | ADCA1 (10-bit resolution) | — | — | √ |
| Port 20 | P20_4 - 5 | 2 bits | In/Out | ISO | | — | √ | — |
| | P20_0 - 5 | 6 bits | | | | — | — | √ |
| Analog Port 0 | AP0_0 - 15 | 16 bits | In/Out | AWO | ADCA0 (12/10-bit resolution) | √ | √ | √ |
| Analog Port 1 | AP1_0 - 7 | 8 bits | In/Out | ISO | ADCA1 (12/10-bit resolution) | — | √ | — |
| | AP1_0 - 15 | 16 bits | | | ADCA1 (12/10-bit resolution) | — | — | √ |
| Input Port 0 | IP0_0 | 1 bit | In | AWO | SOSC (XT2 pin) | — | √ | √ |

## 2.10.1 JTAG Port 0 (JP0)

### 2.10.1.1 Alternative Function

The following alternative functions are available when JTAG port 0 is configured as a general-purpose I/O port by setting OPJTAG[1:0] on the corresponding option byte to $00_B$.

**Table 2.38 JTAG Port 0 (JP0)**

| Port Mode (JPMC0_m = 0) | Alternative Mode (JPMC0_m = 1) | | | | | | | | | | Special Function | PKG No. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1st Alternative | | 2nd Alternative | | 3rd Alternative | | 4th Alternative | | 5th Alternative | | | | | |
| | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | | 100 pins | 144 pins | 176 pins |
| JP0_0*1 | INTP0 | | | | FPDR | FPDT | | | | | DCUTDI/LPDI/ LPDIO | 28 | 39 | 47 |
| JP0_1 | INTP1 | | TAUJ0I0 | TAUJ0O0 | | FPDT | | | | | DCUTDO/LPDO | 27 | 38 | 46 |
| JP0_2 | INTP2 | | TAUJ0I1 | TAUJ0O1 | FPCK | | | | | | DCUTCK/ LPDCLK | 26 | 37 | 45 |
| JP0_3 | INTP3 | CSCXFOUT | TAUJ0I2 | TAUJ0O2 | | | | | | | DCUTMS | 25 | 36 | 44 |
| JP0_4 | | | | | | | | | | | $\overline{\text{DCUTRST}}$ | 24 | 35 | 43 |
| JP0_5 | NMI | RTCA0OUT*2 | TAUJ0I3 | TAUJ0O3 | | | | | | | $\overline{\text{DCURDY}}$/ LPDCLKOUT | 23 | 34 | 42 |
| JP0_6 | | | | | | | | | | | $\overline{\text{EVTO}}$ | — | 54 | 66 |

Note 1. In LPD (1-pin) mode, the JP0_0 output buffer state is Open-drain.

Note 2. Non-existent in 100 pins PKG.

**CAUTION**

The behavior and performance are not guaranteed when alternative functions are not assigned to the register.

### 2.10.1.2 Control Registers

Table 2.39 Control Registers (JP0) (1/2)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| JP0 | JTAG port register 0 | 8 | 5-0 | R/W | $0000_H$ | $00_H$ | √ | — | — |
| | | | 6-0 | | | | — | √ | √ |
| JPSR0 | JTAG port set/reset register 0 | 32 | 21-16, 5-0 | R/W | $0010_H$ | $0000\ 0000_H$ | √ | — | — |
| | | | 22-16, 6-0 | | | | — | √ | √ |
| JPPR0 | JTAG port pin read register 0 | 8 | 5-0 | R | $0020_H$ | $00_H$ | √ | — | — |
| | | | 6-0 | | | | — | √ | √ |
| JPM0 | JTAG port mode register 0 | 8 | 5-0 | R/W | $0030_H$ | $FF_H$ | √ | — | — |
| | | | 6-0 | | | | — | √ | √ |
| JPMC0 | JTAG port mode control register 0 | 8 | 5, 3-0 | R/W | $0040_H$ | $00_H$ | √ | √ | √ |
| JPFC0 | JTAG port function control register 0 | 8 | 5, 3-1 | R/W | $0050_H$ | $00_H$ | √ | √ | √ |
| JPFCE0 | JTAG port function control expansion register 0 | 8 | 2-0 | R/W | $0060_H$ | $00_H$ | √ | √ | √ |
| JPNOT0 | JTAG port NOT register 0 | 8 | 5-0 | W | $0070_H$ | $00_H$ | √ | — | — |
| | | | 6-0 | | | | — | √ | √ |
| JPMSR0 | JTAG port mode set/reset register 0 | 32 | 21-16, 5-0 | R/W | $0080_H$ | $0000\ FFFF_H$ | √ | — | — |
| | | | 22-16, 6-0 | | | | — | √ | √ |
| JPMCSR0 | JTAG port mode control set/reset register 0 | 32 | 21, 19-16, 5, 3-0 | R/W | $0090_H$ | $0000\ 0000_H$ | √ | √ | √ |
| JPIBC0 | JTAG port input buffer control register 0 | 8 | 5-0 | R/W | $0400_H$ | $00_H$ | √ | — | — |
| | | | 6-0 | | | | — | √ | √ |
| JPBDC0 | JTAG port bidirection control register 0 | 8 | 5-0 | R/W | $0410_H$ | $00_H$ | √ | — | — |
| | | | 6-0 | | | | — | √ | √ |
| JPU0 | Pull-up option register 0 | 8 | 5-0 | R/W | $0430_H$ | $00_H$ | √ | — | — |
| | | | 6-0 | | | | — | √ | √ |
| JPD0 | Pull-down option register 0 | 8 | 5-0 | R/W | $0440_H$ | $00_H$ | √ | — | — |
| | | | 6-0 | | | | — | √ | √ |
| JPODC0 | JTAG port open drain control register 0 | 32 | 5-0 | R/W | $0450_H$ | $0000\ 0000_H$ | √ | — | — |
| | | | 6-0 | | | | — | √ | √ |
| JPDSC0 | JTAG port drive strength control register 0 | 32 | 5, 3-1 | R/W | $0460_H$ | $0000\ 0000_H$ | √ | — | — |
| | | | 6, 5, 3-1 | | | | — | √ | √ |
| JPIS0 | JTAG port input buffer selection register 0 | 8 | 5, 3-0 | R/W | $0470_H$ | $FF_H$ | √ | — | — |
| | | | 6, 5, 3-0 | | | | — | √ | √ |

Table 2.39 Control Registers (JP0) (2/2)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| JPISA0 | JTAG port input buffer selection advanced register 0 | 8 | 3, 2, 0 | R/W | $04A0_H$ | $00_H$ | √ | √ | √ |
| JPPROTS0 | JTAG port protection status register 0 | 32 | 0 | R | $04B0_H$ | $0000\ 0000_H$ | √ | √ | √ |
| JPPCMD0 | JTAG port protection command register 0 | 32 | 7-0 | W | $04C0_H$ | $xxxx\ xx00_H$ | √ | √ | √ |

Note 1. The unused bits are read-only (R). When read, the value after reset is returned.
When writing to unused bits, write the value after reset.

## 2.10.2 Port 0 (P0)

### 2.10.2.1 Alternative Function

Table 2.40 Port 0 (P0)

| Port Mode (PMC0_m = 0) | Alternative Mode (PMC0_m =1) | | | | | | | | | | | | | | Special Function | PKG No. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1st Alternative | | 2nd Alternative | | 3rd Alternative | | 4th Alternative | | 5th Alternative | | 6th Alternative | | 7th Alternative | | | 100 pins | 144 pins | 176 pins |
| | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | | | | |
| P0_0 | TAUD0I2 | TAUD0O2 | RLIN20RX | CAN0TX | | PWGA10O | $\overline{\text{CSIH0SSI}}$ | DPO | | | | | | | | 6 | 13 | 18 |
| P0_1 | TAUD0I4 | TAUD0O4 | CAN0RX/ INTP0 | RLIN20TX | INTP0 | PWGA11O | CSIH0SI | APO | | | | | CAN0RX | | | 7 | 14 | 19 |
| P0_2 | TAUD0I6 | TAUD0O6 | CAN1RX/ INTP1 | RLIN30TX | | PWGA12O | CSIH0SC | | INTP1 | DPO | | | CAN1RX | | | 8 | 15 | 20 |
| P0_3 | TAUD0I8 | TAUD0O8 | RLIN30RX/ INTP10 | CAN1TX | DPIN1 | PWGA13O | | CSIH0SO | INTP10 | | | | RLIN30RX | | | 9 | 16 | 21 |
| P0_4 | RLIN31RX/ INTP11 | CAN2TX | INTP11 | PWGA10O | CSIH1SI | SELDP0 | DPIN8 | | | | | | RLIN31RX | | | 11 | 18 | 23 |
| P0_5 | CAN2RX/ INTP2 | RLIN31TX | DPIN9 | SELDP1 | | CSIH1SO | | | | | | | CAN2RX | | | 12 | 19 | 24 |
| P0_6 | INTP2 | | DPIN10 | SELDP2 | CSIH1SC | | | | | PWGA35O | | | | | | 13 | 20 | 25 |
| P0_7 | RLIN21RX | | DPIN5 | CSCXFOUT | CSIH1RYI | CSIH1RYO | TAUB0I0 | TAUB0O0 | CAN3RX/ INTP3 | | | | CAN3RX | | | 40 | 58 | 70 |
| P0_8 | | RLIN21TX | DPIN6 | CSIH0CSS6 | $\overline{\text{CSIH1SSI}}$ | | TAUB0I2 | TAUB0O2 | | CAN3TX | | | | | | 39 | 57 | 69 |
| P0_9 | INTP12 | CSIH1CSS0 | DPIN7 | | RLIN22RX | | TAUB0I4 | TAUB0O4 | CAN4RX/ INTP4 | | | | CAN4RX | | | 38 | 56 | 68 |
| P0_10 | INTP3 | CSIH1CSS1 | DPIN11 | | | RLIN22TX | TAUB0I6 | TAUB0O6 | | CAN4TX | | | | | | 37 | 55 | 67 |
| P0_11 | RIIC0SDA | | DPIN12 | CSIH1CSS2 | TAUB0I8 | TAUB0O8 | RLIN26RX *1 | PWGA34O | | | | | | | | 14 | 21 | 26 |
| P0_12 | RIIC0SCL | | DPIN13 | PWGA45O | TAUB0I10 | TAUB0O10 | CSIG0SI | RLIN26TX *1 | | | | | | | | 15 | 22 | 27 |
| P0_13 | RLIN32RX/ INTP12 | | INTP12 | PWGA46O | TAUB0I12 | TAUB0O12 | | CSIG0SO | CAN5RX/ INTP5 | | RLIN32RX | | CAN5RX | | | 16 | 23 | 28 |
| P0_14 | | RLIN32TX | | PWGA47O | TAUB0I14 | TAUB0O14 | CSIG0SC | | | CAN5TX | | | | | | 17 | 24 | 29 |

Note 1. Non-existent in 100 and 144 pins PKG.

**CAUTION**

The behavior and performance are not guaranteed when alternative functions are not assigned to the register.

### 2.10.2.2 Control Registers

Table 2.41 Control Registers (P0)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| P0 | Port register 0 | 16 | 14-0 | R/W | $0000_H$ | $0000_H$ | √ | √ | √ |
| PSR0 | Port set/reset register 0 | 32 | 30-16, 14-0 | R/W | $0100_H$ | $0000\ 0000_H$ | √ | √ | √ |
| PPR0 | Port pin read register 0 | 16 | 14-0 | R | $0200_H$ | $0000_H$ | √ | √ | √ |
| PM0 | Port mode register 0 | 16 | 14-0 | R/W | $0300_H$ | $FFFF_H$ | √ | √ | √ |
| PMC0 | Port mode control register 0 | 16 | 14-0 | R/W | $0400_H$ | $0000_H$ | √ | √ | √ |
| PFC0 | Port function control register 0 | 16 | 14-0 | R/W | $0500_H$ | $0000_H$ | √ | √ | √ |
| PFCE0 | Port function control expansion register 0 | 16 | 14-0 | R/W | $0600_H$ | $0000_H$ | √ | √ | √ |
| PNOT0 | Port NOT register 0 | 16 | 14-0 | W | $0700_H$ | $0000_H$ | √ | √ | √ |
| PMSR0 | Port mode set/reset register 0 | 32 | 30-16, 14-0 | R/W | $0800_H$ | $0000\ FFFF_H$ | √ | √ | √ |
| PMCSR0 | Port mode control set/reset register 0 | 32 | 30-16, 14-0 | R/W | $0900_H$ | $0000\ 0000_H$ | √ | √ | √ |
| PFCAE0 | Port function control additional expansion register 0 | 16 | 14, 13, 10-1 | R/W | $0A00_H$ | $0000_H$ | √ | √ | √ |
| PIBC0 | Port input buffer control register 0 | 16 | 14-0 | R/W | $4000_H$ | $0000_H$ | √ | √ | √ |
| PBDC0 | Port bidirection control register 0 | 16 | 14-0 | R/W | $4100_H$ | $0000_H$ | √ | √ | √ |
| PIPC0 | Port IP control register 0 | 16 | 14, 13, 6, 5, 3, 2 | R/W | $4200_H$ | $0000_H$ | √ | √ | √ |
| PU0 | Pull-up option register 0 | 16 | 14-0 | R/W | $4300_H$ | $0000_H$ | √ | √ | √ |
| PD0 | Pull-down option register 0 | 16 | 14-0 | R/W | $4400_H$ | $0000_H$ | √ | √ | √ |
| PODC0 | Port open drain control register 0 | 32 | 14-0 | R/W | $4500_H$ | $0000\ 0000_H$ | √ | √ | √ |
| PDSC0 | Port drive strength control register 0 | 32 | 14-0 | R/W | $4600_H$ | $0000\ 0000_H$ | √ | √ | √ |
| PIS0 | Port input buffer selection register 0 | 16 | 14-0 | R/W | $4700_H$ | $FFFF_H$ | √ | √ | √ |
| PPROTS0 | Port protection status register 0 | 32 | 0 | R | $4B00_H$ | $0000\ 0000_H$ | √ | √ | √ |
| PPCMD0 | Port protection command register 0 | 32 | 7-0 | W | $4C00_H$ | $XXXX\ XX00_H$ | √ | √ | √ |

Note 1. The unused bits are read-only (R). When read, the value after reset is returned.
When writing to unused bits, write the value after reset.

## 2.10.3 Port 1 (P1)

### 2.10.3.1 Alternative Function

Table 2.42 Port 1 (P1)

| Port Mode (PMC1_m = 0) | Alternative Mode (PMC1_m = 1) | | | | | | | | | | | | | | Special Function | PKG No. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1st Alternative | | 2nd Alternative | | 3rd Alternative | | 4th Alternative | | 5th Alternative | | 6th Alternative | | 7th Alternative | | | 100 pins | 144 pins | 176 pins |
| | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | | | | |
| P1_0 | RLIN33RX/ INTP13 | | INTP13 | | | | | | | | | | RLIN33RX | | | — | 25 | 30 |
| P1_1 | | RLIN33TX | | | | | | | | | | | | | | — | 26 | 31 |
| P1_2 | CAN3RX/ INTP3 | | INTP3 | | | | | | | | | | CAN3RX | | | — | 27 | 32 |
| P1_3 | | CAN3TX | DPIN23 | | | | | | | | | | | | | — | 28 | 33 |
| P1_4 | RLIN35RX/ INTP15 | | INTP15 | | | | | | | | | | RLIN35RX | | | — | 63 | 75 |
| P1_5 | ADCA1TRG0 | RLIN35TX | DPIN17 | | | | | | | | | | | | | — | 62 | 74 |
| P1_6 | RLIN25RX | | DPIN18 | | | | | | | | | | | | | — | 61 | 73 |
| P1_7 | ADCA1TRG1 | RLIN25TX | DPIN19 | | | | | | | | | | | | | — | 60 | 72 |
| P1_8 | RLIN34RX/ INTP14 | | INTP14 | | | | | | | | | | RLIN34RX | | | — | 43 | 53 |
| P1_9 | | RLIN34TX | DPIN20 | | | | | | | | | | | | | — | 42 | 52 |
| P1_10 | RLIN24RX | | DPIN21 | | | | | | | | | | | | | — | 41 | 51 |
| P1_11 | ADCA1TRG2 | RLIN24TX | DPIN22 | | | | | | | | | | | | | — | 40 | 50 |
| P1_12 | CAN4RX/ INTP4 | | INTP4 | | | | | | | | | | CAN4RX | | | — | — | 34 |
| P1_13 | | CAN4TX | | | | | | | | | | | | | | — | — | 35 |
| P1_14 | RLIN23RX | | | | | | | | | | | | | | | — | — | 78 |
| P1_15 | | RLIN23TX | | | | | | | | | | | | | | — | — | 79 |

**CAUTION**

The behavior and performance are not guaranteed when alternative functions are not assigned to the register.

### 2.10.3.2 Control Registers

Table 2.43 Control Registers (P1) (1/2)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| P1 | Port register 1 | 16 | 11-0 | R/W | $0004_H$ | $0000_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| PSR1 | Port set/reset register 1 | 32 | 27-16, 11-0 | R/W | $0104_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 31-16, 15-0 | | | | — | — | √ |
| PPR1 | Port pin read register 1 | 16 | 11-0 | R | $0204_H$ | $0000_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| PM1 | Port mode register 1 | 16 | 11-0 | R/W | $0304_H$ | $FFFF_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| PMC1 | Port mode control register 1 | 16 | 11-0 | R/W | $0404_H$ | $0000_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| PFC1 | Port function control register 1 | 16 | 11-2, 0 | R/W | $0504_H$ | $0000_H$ | — | √ | — |
| | | | 12-2, 0 | | | | — | — | √ |
| PFCE1 | Port function control expansion register 1 | 16 | 8, 4, 2, 0 | R/W | $0604_H$ | $0000_H$ | — | √ | — |
| | | | 12, 8, 4, 2, 0 | | | | — | — | √ |
| PNOT1 | Port NOT register 1 | 16 | 11-0 | W | $0704_H$ | $0000_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| PMSR1 | Port mode set/reset register 1 | 32 | 27-16, 11-0 | R/W | $0804_H$ | $0000\ FFFF_H$ | — | √ | — |
| | | | 31-16, 15-0 | | | | — | — | √ |
| PMCSR1 | Port mode control set/reset register 1 | 32 | 27-16, 11-0 | R/W | $0904_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 31-16, 15-0 | | | | — | — | √ |
| PFCAE1 | Port function control additional expansion register 1 | 16 | 8, 4, 2, 0 | R/W | $0A04_H$ | $0000_H$ | — | √ | — |
| | | | 12, 8, 4, 2, 0 | | | | — | — | √ |
| PIBC1 | Port input buffer control register 1 | 16 | 11-0 | R/W | $4004_H$ | $0000_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| PBDC1 | Port bidirection control register 1 | 16 | 11-0 | R/W | $4104_H$ | $0000_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| PU1 | Pull-up option register 1 | 16 | 11-0 | R/W | $4304_H$ | $0000_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| PD1 | Pull-down option register 1 | 16 | 11-0 | R/W | $4404_H$ | $0000_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| PODC1 | Port open drain control register 1 | 32 | 11-0 | R/W | $4504_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| PDSC1 | Port drive strength control register 1 | 32 | 11-0 | R/W | $4604_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |

Table 2.43 Control Registers (P1) (2/2)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| PIS1 | Port input buffer selection register 1 | 16 | 11-0 | R/W | $4704_H$ | $FFFF_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| PPROTS1 | Port protection status register 1 | 32 | 0 | R | $4B04_H$ | $0000\ 0000_H$ | — | √ | √ |
| PPCMD1 | Port protection command register 1 | 32 | 7-0 | W | $4C04_H$ | $xxxx\ xx00_H$ | — | √ | √ |

Note 1. The unused bits are read-only (R). When read, the value after reset is returned.
When writing to unused bits, write the value after reset.

## 2.10.4 Port 2 (P2)

### 2.10.4.1 Alternative Function

Table 2.44 Port 2 (P2)

| Port Mode (PMC2_m =0) | Alternative Mode (PMC2_m =1) | | | | | | | | | | | | | | Special Function | PKG No. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1st Alternative | | 2nd Alternative | | 3rd Alternative | | 4th Alternative | | 5th Alternative | | 6th Alternative | | 7th Alternative | | | 100 pins | 144 pins | 176 pins |
| | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | | | | |
| P2_0 | RLIN27RX | | INTP6/CAN6RX | | | | | | | | | | CAN6RX | | | — | — | 49 |
| P2_1 | | RLIN27TX | | CAN6TX | | | | | | | | | | | | — | — | 48 |
| P2_2 | RLIN28RX | | | | | | | | | | | | | | | — | — | 65 |
| P2_3 | | RLIN28TX | | | | | | | | | | | | | | — | — | 64 |
| P2_4 | RLIN29RX | | | ADCA0SEL0 | | | | | | | | | | | | — | — | 76 |
| P2_5 | | RLIN29TX | | ADCA0SEL1 | | | | | | | | | | | | — | — | 77 |
| P2_6 | | ADCA0SEL2 | | | | | | | | | | | | | | — | — | 36 |

**CAUTION**

The behavior and performance are not guaranteed when alternative functions are not assigned to the register.

### 2.10.4.2 Control Registers

Table 2.45 Control Registers (P2)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| P2 | Port register 2 | 16 | 6-0 | R/W | $0008_H$ | $0000_H$ | — | — | √ |
| PSR2 | Port set/reset register 2 | 32 | 22-16, 6-0 | R/W | $0108_H$ | 0000 $0000_H$ | — | — | √ |
| PPR2 | Port pin read register 2 | 16 | 6-0 | R | $0208_H$ | $0000_H$ | — | — | √ |
| PM2 | Port mode register 2 | 16 | 6-0 | R/W | $0308_H$ | $FFFF_H$ | — | — | √ |
| PMC2 | Port mode control register 2 | 16 | 6-0 | R/W | $0408_H$ | $0000_H$ | — | — | √ |
| PFC2 | Port function control register 2 | 16 | 5, 4, 1, 0 | R/W | $0508_H$ | $0000_H$ | — | — | √ |
| PFCE2 | Port function control expansion register 2 | 16 | 0 | R/W | $0608_H$ | $0000_H$ | — | — | √ |
| PNOT2 | Port NOT register 2 | 16 | 6-0 | W | $0708_H$ | $0000_H$ | — | — | √ |
| PMSR2 | Port mode set/reset register 2 | 32 | 22-16, 6-0 | R/W | $0808_H$ | 0000 $FFFF_H$ | — | — | √ |
| PMCSR2 | Port mode control set/reset register 2 | 32 | 22-16, 6-0 | R/W | $0908_H$ | 0000 $0000_H$ | — | — | √ |
| PFCAE2 | Port function control additional expansion register 2 | 16 | 0 | R/W | $0A08_H$ | $0000_H$ | — | — | √ |
| PIBC2 | Port input buffer control register 2 | 16 | 6-0 | R/W | $4008_H$ | $0000_H$ | — | — | √ |
| PBDC2 | Port bidirection control register 2 | 16 | 6-0 | R/W | $4108_H$ | $0000_H$ | — | — | √ |
| PU2 | Pull-up option register 2 | 16 | 6-0 | R/W | $4308_H$ | $0000_H$ | — | — | √ |
| PD2 | Pull-down option register 2 | 16 | 6-0 | R/W | $4408_H$ | $0000_H$ | — | — | √ |
| PODC2 | Port open drain control register 2 | 32 | 6-0 | R/W | $4508_H$ | 0000 $0000_H$ | — | — | √ |
| PDSC2 | Port drive strength control register 2 | 32 | 6-0 | R/W | $4608_H$ | 0000 $0000_H$ | — | — | √ |
| PIS2 | Port input buffer selection register 2 | 16 | 6-0 | R/W | $4708_H$ | $FFFF_H$ | — | — | √ |
| PPROTS2 | Port protection status register 2 | 32 | 0 | R | $4B08_H$ | 0000 $0000_H$ | — | — | √ |
| PPCMD2 | Port protection command register 2 | 32 | 7-0 | W | $4C08_H$ | xxxx $xx00_H$ | — | — | √ |

Note 1. The unused bits are read-only (R). When read, the value after reset is returned.
When writing to unused bits, write the value after reset.

## 2.10.5 Port 8 (P8)

### 2.10.5.1 Alternative Function

Table 2.46 Port 8 (P8)

| Port Mode (PMC8_m = 0) | Alternative Mode (PMC8_m = 1) | | | | | | | | | | ADC | Special Function | PKG No. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1st Alternative | | 2nd Alternative | | 3rd Alternative | | 4th Alternative | | 5th Alternative | | | | 100 pins | 144 pins | 176 pins |
| | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | | | | | |
| P8_0 | TAUJ0I0 | TAUJ0O0 | DPIN2 | PWGA14O | INTP4 | CSIH0CSS0 | | | | | ADCA0I0S | | 42 | 64 | 80 |
| P8_1 | TAPA0ESO | TAUJ0O1 | DPIN0 | PWGA15O | INTP5 | CSIH1CSS3 | | | | | ADCA0I1S | | 43 | 65 | 81 |
| P8_2 | TAUJ0I0 | TAUJ0O0 | DPIN2 | CSIH0CSS0 | INTP6 | PWGA22O | | | | | ADCA0I4S | | 19 | 30 | 38 |
| P8_3 | TAUJ0I1 | TAUJ0O1 | DPIN3 | CSIH0CSS1 | INTP7 | PWGA23O | | | | | ADCA0I5S | | 44 | 66 | 82 |
| P8_4 | TAUJ0I2 | TAUJ0O2 | DPIN4 | CSIH0CSS2 | INTP8 | PWGA36O | | | | | ADCA0I6S | | 45 | 67 | 83 |
| P8_5 | TAUJ0I3 | TAUJ0O3 | NMI | CSIH0CSS3 | INTP9*1 | PWGA37O | | | | | ADCA0I7S | | 46 | 68 | 84 |
| P8_6 | NMI | CSIH0CSS4 | | PWGA38O | | RTCA0OUT*1 | | | | | ADCA0I8S | $\overline{\text{RESETOUT}}$ | 47 | 69 | 85 |
| P8_7 | | CSIH3CSS0 | | PWGA39O | | ADCA0SEL0 | | RTCA0OUT | | | ADCA0I14S | | 48 | 70 | 86 |
| P8_8 | | CSIH3CSS1 | | PWGA40O | | ADCA0SEL1 | | | | | ADCA0I15S | | 49 | 71 | 87 |
| P8_9 | | CSIH3CSS2 | | PWGA41O | | ADCA0SEL2 | | | | | ADCA0I16S | | 50 | 72 | 88 |
| P8_10 | | CSIH3CSS3 | DPIN14 | PWGA42O | | | | | | | ADCA0I17S | | 20 | 31 | 39 |
| P8_11 | TAUJ1I2 | TAUJ1O2 | DPIN15 | PWGA43O | | CSIH1CSS4 | | | | | ADCA0I18S | | 21 | 32 | 40 |
| P8_12 | TAUJ1I3 | TAUJ1O3 | DPIN16 | PWGA44O | | CSIH1CSS5 | | | | | ADCA0I19S | | 22 | 33 | 41 |

Note 1. Non-existent in 100 pins PKG.

**CAUTIONS**

1. The behavior and performance are not guaranteed when alternative functions are not assigned to the register.
2. Use ADC functions with their initial settings.
3. When the $\overline{\text{RESETOUT}}$ function is selected for the P8_6 pin, the P8_6 pin outputs a low-level as the $\overline{\text{RESETOUT}}$ signal while a reset is asserted and continues to output a low level after the reset is released. For details, see **Section 2.11.1.1, P8_6: RESETOUT**.

### 2.10.5.2 Control Registers

Table 2.47 Control Registers (P8)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| P8 | Port register 8 | 16 | 12-0 | R/W | $0020_H$ | $0000_H$ | √ | √ | √ |
| PSR8 | Port set/reset register 8 | 32 | 28-16, 12-0 | R/W | $0120_H$ | $0000\ 0000_H$ | √ | √ | √ |
| PPR8 | Port pin read register 8 | 16 | 12-0 | R | $0220_H$ | $0000_H$ | √ | √ | √ |
| PM8 | Port mode register 8 | 16 | 12-0 | R/W | $0320_H$ | $FFBF_H$ | √ | √ | √ |
| PMC8 | Port mode control register 8 | 16 | 12-0 | R/W | $0420_H$ | $0000_H$ | √ | √ | √ |
| PFC8 | Port function control register 8 | 16 | 12-0 | R/W | $0520_H$ | $0000_H$ | √ | √ | √ |
| PFCE8 | Port function control expansion register 8 | 16 | 12, 11, 9-7, 5-0 | R/W | $0620_H$ | $0000_H$ | √ | — | — |
| | | | 12, 11, 9-0 | | | | — | √ | √ |
| PNOT8 | Port NOT register 8 | 16 | 12-0 | W | $0720_H$ | $0000_H$ | √ | √ | √ |
| PMSR8 | Port mode set/reset register 8 | 32 | 28-16, 12-0 | R/W | $0820_H$ | $0000\ FFBF_H$ | √ | √ | √ |
| PMCSR8 | Port mode control set/reset register 8 | 32 | 28-16, 12-0 | R/W | $0920_H$ | $0000\ 0000_H$ | √ | √ | √ |
| PIBC8 | Port input buffer control register 8 | 16 | 12-0 | R/W | $4020_H$ | $0000_H$ | √ | √ | √ |
| PBDC8 | Port bidirection control register 8 | 16 | 12-0 | R/W | $4120_H$ | $0000_H$ | √ | √ | √ |
| PU8 | Pull-up option register 8 | 16 | 12-0 | R/W | $4320_H$ | $0000_H$ | √ | √ | √ |
| PD8 | Pull-down option register 8 | 16 | 12-0 | R/W | $4420_H$ | $0000_H$ | √ | √ | √ |
| PODC8 | Port open drain control register 8 | 32 | 12-0 | R/W | $4520_H$ | $0000\ 0040_H$ | √ | √ | √ |
| PIS8 | Port input buffer selection register 8 | 32 | 12-0 | R/W | $4720_H$ | $FFFF_H$ | √ | √ | √ |
| PPROTS8 | Port protection status register 8 | 32 | 0 | R | $4B20_H$ | $0000\ 0000_H$ | √ | √ | √ |
| PPCMD8 | Port protection command register 8 | 32 | 7-0 | W | $4C20_H$ | $xxxx\ xx00_H$ | √ | √ | √ |

Note 1. The unused bits are read-only (R). When read, the value after reset is returned.
When writing to unused bits, write the value after reset.

**CAUTION**

**P8_6 drives a low level after any kind of reset release, until it is later configured differently by register settings.**

**To avoid a data collision, the outside circuit connected to this pin must not drive a high level.**

## 2.10.6 Port 9 (P9)

### 2.10.6.1 Alternative Function

Table 2.48 Port 9 (P9)

| Port Mode (PMC9_m = 0) | Alternative Mode (PMC9_m = 1) | | | | | | | | | | ADC | PKG No. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1st Alternative | | 2nd Alternative | | 3rd Alternative | | 4th Alternative | | 5th Alternative | | | | | |
| | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | | 100 pins | 144 pins | 176 pins |
| P9_0 | NMI | PWGA8O | TAUD0I0 | TAUD0O0 | ADCA0TRG0 | CSIH2CSS0 | KR0I4 | | | | ADCA0I2S | 69 | 92 | 108 |
| P9_1 | INTP11 | PWGA9O | TAUD0I2 | TAUD0O2 | KR0I5 | CSIH2CSS1 | | | | | ADCA0I3S | 70 | 93 | 109 |
| P9_2 | KR0I6 | PWGA20O | TAPA0ESO | CSIH2CSS2 | | | | | | | ADCA0I9S | 71 | 94 | 110 |
| P9_3 | KR0I7 | PWGA21O | | CSIH2CSS3 | TAUJ1I1 | TAUJ1O1 | | | | | ADCA0I10S | 72 | 95 | 111 |
| P9_4 | | CSIH0CSS5 | | PWGA33O | TAUJ1I0 | TAUJ1O0 | | | | | ADCA0I11S | 73 | 96 | 112 |
| P9_5 | | CSIH0CSS6 | | PWGA34O | TAUJ1I1 | TAUJ1O1 | | | | | ADCA0I12S | 74 | 97 | 113 |
| P9_6 | | CSIH0CSS7 | | PWGA35O | | | | | | | ADCA0I13S | 75 | 98 | 114 |

**CAUTIONS**

1. The behavior and performance are not guaranteed when alternative functions are not assigned to the register.
2. Use ADC functions with their initial settings.

### 2.10.6.2 Control Registers

Table 2.49 Control Registers (P9)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| P9 | Port register 9 | 16 | 6-0 | R/W | $0024_H$ | $0000_H$ | √ | √ | √ |
| PSR9 | Port set/reset register 9 | 32 | 22-16, 6-0 | R/W | $0124_H$ | $0000\ 0000_H$ | √ | √ | √ |
| PPR9 | Port pin read register 9 | 16 | 6-0 | R | $0224_H$ | $0000_H$ | √ | √ | √ |
| PM9 | Port mode register 9 | 16 | 6-0 | R/W | $0324_H$ | $FFFF_H$ | √ | √ | √ |
| PMC9 | Port mode control register 9 | 16 | 6-0 | R/W | $0424_H$ | $0000_H$ | √ | √ | √ |
| PFC9 | Port function control register 9 | 16 | 6-0 | R/W | $0524_H$ | $0000_H$ | √ | √ | √ |
| PFCE9 | Port function control expansion register 9 | 16 | 5-3, 1, 0 | R/W | $0624_H$ | $0000_H$ | √ | √ | √ |
| PNOT9 | Port NOT register 9 | 16 | 6-0 | W | $0724_H$ | $0000_H$ | √ | √ | √ |
| PMSR9 | Port mode set/reset register 9 | 32 | 22-16, 6-0 | R/W | $0824_H$ | $0000\ FFFF_H$ | √ | √ | √ |
| PMCSR9 | Port mode control set/reset register 9 | 32 | 22-16, 6-0 | R/W | $0924_H$ | $0000\ 0000_H$ | √ | √ | √ |
| PIBC9 | Port input buffer control register 9 | 16 | 6-0 | R/W | $4024_H$ | $0000_H$ | √ | √ | √ |
| PBDC9 | Port bidirection control register 9 | 16 | 6-0 | R/W | $4124_H$ | $0000_H$ | √ | √ | √ |
| PU9 | Pull-up option register 9 | 16 | 6-0 | R/W | $4324_H$ | $0000_H$ | √ | √ | √ |
| PD9 | Pull-down option register 9 | 16 | 6-0 | R/W | $4424_H$ | $0000_H$ | √ | √ | √ |
| PODC9 | Port open drain control register 9 | 32 | 6-0 | R/W | $4524_H$ | $0000\ 0000_H$ | √ | √ | √ |
| PIS9 | Port input buffer selection register 9 | 16 | 6-0 | R/W | $4724_H$ | $FFFF_H$ | √ | √ | √ |
| PPROTS9 | Port protection status register 9 | 32 | 0 | R | $4B24_H$ | $0000\ 0000_H$ | √ | √ | √ |
| PPCMD9 | Port protection command register 9 | 32 | 7-0 | W | $4C24_H$ | $xxxx\ xx00_H$ | √ | √ | √ |

Note 1. The unused bits are read-only (R). When read, the value after reset is returned.
When writing to unused bits, write the value after reset.

## 2.10.7 Port 10 (P10)

### 2.10.7.1 Alternative Function

Table 2.50 Port 10 (P10)

| Port Mode (PMC10_m = 0) | Alternative Mode (PMC10_m = 1) | | | | | | | | | | | | | | Special Function | PKG No. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1st Alternative | | 2nd Alternative | | 3rd Alternative | | 4th Alternative | | 5th Alternative | | 6th Alternative | | 7th Alternative | | | 100 pins | 144 pins | 176 pins |
| | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | | | | |
| P10_0 | TAUD0I1 | TAUD0O1 | CAN0RX/ INTP0 | CSCXFOUT | | PWGA0O | | TAPA0UP | CSIH1SI | | | | CAN0RX | | | 98 | 142 | 174 |
| P10_1 | TAUD0I3 | TAUD0O3 | | CAN0TX | | PWGA1O | | TAPA0UN | CSIH1SC | | | | | | MODE0 | 99 | 143 | 175 |
| P10_2 | TAUD0I5 | TAUD0O5 | RIIC0SDA | | KR0I0 | PWGA2O | ADCA0TRG0 | TAPA0VP | | CSIH1SO | | | | | MODE1 | 100 | 144 | 176 |
| P10_3 | TAUD0I7 | TAUD0O7 | RIIC0SCL | | KR0I1 | PWGA3O | ADCA0TRG1 | TAPA0VN | $\overline{\text{CSIH1SSI}}$ | | | | | | | 1 | 1 | 1 |
| P10_4 | TAUD0I9 | TAUD0O9 | RLIN21RX | | KR0I2 | ADCA0SEL0 | ADCA0TRG2 | TAPA0WP | $\overline{\text{CSIG0SSI}}$ | | | | | | | 2 | 2 | 2 |
| P10_5 | TAUD0I11 | TAUD0O11 | | RLIN21TX | KR0I3 | ADCA0SEL1 | | TAPA0WN | CSIG0RYI | CSIG0RYO | | | | | | 3 | 3 | 3 |
| P10_6 | TAUD0I13 | TAUD0O13 | | CSIG0SO | ENCA0TIN0 | ADCA0SEL2 | CAN1RX /INTP1 | | | | | | CAN1RX | | MODE2 | 80 | 120 | 152 |
| P10_7 | TAUD0I15 | TAUD0O15 | CSIG0SC | | ENCA0TIN1 | PWGA4O | | CAN1TX | | | | | | | | 81 | 121 | 153 |
| P10_8 | TAUD0I10 | TAUD0O10 | CSIG0SI | | ENCA0EC | PWGA5O | | | | | | | | | FLMD1 | 82 | 122 | 154 |
| P10_9 | TAUD0I12 | TAUD0O12 | RLIN30RX /INTP10 | | ENCA0E0 | PWGA6O | CSIH0RYI | CSIH0RYO | | | | | RLIN30RX | | | 83 | 123 | 155 |
| P10_10 | TAUD0I14 | TAUD0O14 | | RLIN30TX | ENCA0E1 | PWGA7O | | CSIH0CSS1 | | | | | | | | 84 | 124 | 156 |
| P10_11 | | PWGA16O | RLIN31RX /INTP11 | | | CSIH1CSS0 | TAUB0I1 | TAUB0O1 | | | | | RLIN31RX | | | 85 | 125 | 157 |
| P10_12 | | PWGA17O | | RLIN31TX | | CSIH1CSS1 | TAUB0I3 | TAUB0O3 | | | | | | | | 86 | 126 | 158 |
| P10_13 | $\overline{\text{CSIH0SSI}}$ | PWGA18O | RLIN32RX /INTP12 | | | | TAUB0I5 | TAUB0O5 | | | | | RLIN32RX | | | 87 | 127 | 159 |
| P10_14 | ADCA1TRG0 *1 | PWGA19O | | RLIN32TX | $\overline{\text{CSIH3SSI}}$ | | TAUB0I7 | TAUB0O7 | | | | | | | | 88 | 128 | 160 |
| P10_15 | CSIH3RYI | CSIH3RYO | | PWGA24O | RLIN22RX | | TAUB0I9 | TAUB0O9 | | | | | | | | 4 | 4 | 6 |

Note 1. Non-existent in 100 pins PKG.

**CAUTION**

The behavior and performance are not guaranteed when alternative functions are not assigned to the register.

### 2.10.7.2 Control Registers

Table 2.51 Control Registers (P10)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| P10 | Port register 10 | 16 | 15-0 | R/W | $0028_H$ | $0000_H$ | √ | √ | √ |
| PSR10 | Port set/reset register 10 | 32 | 31-16, 15-0 | R/W | $0128_H$ | 0000 $0000_H$ | √ | √ | √ |
| PPR10 | Port pin read register 10 | 16 | 15-0 | R | $0228_H$ | $0000_H$ | √ | √ | √ |
| PM10 | Port mode register 10 | 16 | 15-0 | R/W | $0328_H$ | $FFFF_H$ | √ | √ | √ |
| PMC10 | Port mode control register 10 | 16 | 15-0 | R/W | $0428_H$ | $0000_H$ | √ | √ | √ |
| PFC10 | Port function control register 10 | 16 | 15-0 | R/W | $0528_H$ | $0000_H$ | √ | √ | √ |
| PFCE10 | Port function control expansion register 10 | 16 | 15-0 | R/W | $0628_H$ | $0000_H$ | √ | √ | √ |
| PNOT10 | Port NOT register 10 | 16 | 15-0 | W | $0728_H$ | $0000_H$ | √ | √ | √ |
| PMSR10 | Port mode set/reset register 10 | 32 | 31-16, 15-0 | R/W | $0828_H$ | 0000 $FFFF_H$ | √ | √ | √ |
| PMCSR10 | Port mode control set/reset register 10 | 32 | 31-16, 15-0 | R/W | $0928_H$ | 0000 $0000_H$ | √ | √ | √ |
| PFCAE10 | Port function control additional expansion register 10 | 16 | 13, 11, 9, 6-0 | R/W | $0A28_H$ | $0000_H$ | √ | √ | √ |
| PIBC10 | Port input buffer control register 10 | 16 | 15-0 | R/W | $4028_H$ | $0000_H$ | √ | √ | √ |
| PBDC10 | Port bidirection control register 10 | 16 | 15-0 | R/W | $4128_H$ | $0000_H$ | √ | √ | √ |
| PIPC10 | Port IP control register 10 | 16 | 7-0 | R/W | $4228_H$ | $0000_H$ | √ | √ | √ |
| PU10 | Pull-up option register 10 | 16 | 15-0 | R/W | $4328_H$ | $0000_H$ | √ | √ | √ |
| PD10 | Pull-down option register 10 | 16 | 15-0 | R/W | $4428_H$ | $0000_H$ | √ | √ | √ |
| PODC10 | Port open drain control register 10 | 32 | 15-0 | R/W | $4528_H$ | 0000 $0000_H$ | √ | √ | √ |
| PDSC10 | Port drive strength control register 10 | 32 | 15-0 | R/W | $4628_H$ | 0000 $0000_H$ | √ | √ | √ |
| PIS10 | Port input buffer selection register 10 | 16 | 15-0 | R/W | $4728_H$ | $FFFF_H$ | √ | √ | √ |
| PPROTS10 | Port protection status register 10 | 32 | 0 | R | $4B28_H$ | 0000 $0000_H$ | √ | √ | √ |
| PPCMD10 | Port protection command register 10 | 32 | 7-0 | W | $4C28_H$ | xxxx $xx00_H$ | √ | √ | √ |

Note 1. The unused bits are read-only (R). When read, the value after reset is returned.
When writing to unused bits, write the value after reset.

## 2.10.8 Port 11 (P11)

### 2.10.8.1 Alternative Function

Table 2.52 Port 11 (P11)

| Port mode (PMC11_m = 0) | Alternative Mode (PMC11_m = 1) | | | | | | | | | | | | | | Special Function | PKG No. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1st Alternative | | 2nd Alternative | | 3rd Alternative | | 4th Alternative | | 5th Alternative | | 6th Alternative | | 7th Alternative | | | 100 pins | 144 pins | 176 pins |
| | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | | | | |
| P11_0 | CSIH2RYI | CSIH2RYO | ADCA1TRG2*1 | PWGA25O | | RLIN22TX | TAUB0I11 | TAUB0O11 | | | | | | | | 5 | 5 | 7 |
| P11_1 | $\overline{\text{CSIH2SSI}}$ | | RLIN20RX | CSIH0CSS7 | | PWGA26O | TAUB0I13 | TAUB0O13 | | | | | | | | 89 | 129 | 161 |
| P11_2 | | CSIH2SO | RLIN32RX /INTP12 | RLIN20TX | | PWGA27O | TAUB0I15 | TAUB0O15 | | | | | RLIN32RX | | | 90 | 130 | 162 |
| P11_3 | CSIH2SC | | CAN3RX /INTP3 | PWGA28O | TAUB1I1*2 | TAUB1O1*2 | | | | RLIN32TX | | | CAN3RX | | | 91 | 131 | 163 |
| P11_4 | CSIH2SI | CAN3TX | | PWGA29O | TAUB1I3*2 | TAUB1O3*2 | | | | | | | | | | 92 | 132 | 164 |
| P11_5 | CAN5RX/ INTP5 | RLIN33TX | | PWGA30O | CSIH3SI | | TAUB1I5*2 | TAUB1O5*2 | | | | | CAN5RX | | | 93 | 133 | 165 |
| P11_6 | RLIN33RX /INTP13 | CAN5TX | ADCA1TRG1 | PWGA31O | | CSIH3SO | TAUB1I7*2 | TAUB1O7*2 | | | | | RLIN33RX | | | 94 | 134 | 166 |
| P11_7 | INTP5 | | | PWGA32O | CSIH3SC | | TAUB1I9*2 | TAUB1O9*2 | | | | | | | | 95 | 135 | 167 |
| P11_8 | $\overline{\text{CSIG1SSI}}$ | RLIN35TX | | PWGA48O | TAUB1I11*3 | TAUB1O11*3 | | | | | | | | | | — | 6 | 8 |
| P11_9 | | CSIG1SO | RLIN35RX /INTP15 | PWGA49O | TAUB1I13*3 | TAUB1O13*3 | | | | | | | RLIN35RX | | | — | 7 | 9 |
| P11_10 | CSIG1SC | | | PWGA50O | TAUB1I15*3 | TAUB1O15*3 | | | | | | | | | | — | 8 | 10 |
| P11_11 | CSIG1SI | RLIN25TX | | PWGA51O | TAUB1I0*3 | TAUB1O0*3 | | | | | | | | | | — | 9 | 11 |
| P11_12 | RLIN25RX | | | PWGA52O | TAUB1I2*3 | TAUB1O2*3 | | | | | | | | | | — | 10 | 12 |
| P11_13 | RLIN24RX | | | PWGA53O | TAUB1I4*3 | TAUB1O4*3 | | | | | | | | | | — | 11 | 13 |
| P11_14 | | RLIN24TX | | PWGA54O | TAUB1I6*3 | TAUB1O6*3 | | | | | | | | | | — | 12 | 14 |
| P11_15 | CAN2RX /INTP2 | CSIH2CSS4 | | PWGA55O | TAUB1I8*3 | TAUB1O8*3 | | | | | | | CAN2RX | | | — | 136 | 168 |

Note 1. Non-existent in 100 pins PKG.

Note 2. Non-existent in 100 and 144 pins PKG.

Note 3. Non-existent in 144 pins PKG.

**CAUTION**

The behavior and performance are not guaranteed when alternative functions are not assigned to the register.

### 2.10.8.2 Control Registers

Table 2.53 Control Registers (P11) (1/2)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| P11 | Port register 11 | 16 | 7-0 | R/W | $002C_H$ | $0000_H$ | √ | — | — |
| | | | 15-0 | | | | — | √ | √ |
| PSR11 | Port set/reset register 11 | 32 | 23-16, 7-0 | R/W | $012C_H$ | $0000\ 0000_H$ | √ | — | — |
| | | | 31-16, 15-0 | | | | — | √ | √ |
| PPR11 | Port pin read register 11 | 16 | 7-0 | R | $022C_H$ | $0000_H$ | √ | — | — |
| | | | 15-0 | | | | — | √ | √ |
| PM11 | Port mode register 11 | 16 | 7-0 | R/W | $032C_H$ | $FFFF_H$ | √ | — | — |
| | | | 15-0 | | | | — | √ | √ |
| PMC11 | Port mode control register 11 | 16 | 7-0 | R/W | $042C_H$ | $0000_H$ | √ | — | — |
| | | | 15-0 | | | | — | √ | √ |
| PFC11 | Port function control register 11 | 16 | 7-0 | R/W | $052C_H$ | $0000_H$ | √ | — | — |
| | | | 15-0 | | | | — | √ | √ |
| PFCE11 | Port function control expansion register 11 | 16 | 7-5, 3-0 | R/W | $062C_H$ | $0000_H$ | √ | — | — |
| | | | 15, 9, 7-5, 3-0 | | | | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| PNOT11 | Port NOT register 11 | 16 | 7-0 | W | $072C_H$ | $0000_H$ | √ | — | — |
| | | | 15-0 | | | | — | √ | √ |
| PMSR11 | Port mode set/reset register 11 | 32 | 23-16, 7-0 | R/W | $082C_H$ | $0000\ FFFF_H$ | √ | — | — |
| | | | 31-16, 15-0 | | | | — | √ | √ |
| PMCSR11 | Port mode control set/reset register 11 | 32 | 23-16, 7-0 | R/W | $092C_H$ | $0000\ 0000_H$ | √ | — | — |
| | | | 31-16, 15-0 | | | | — | √ | √ |
| PFCAE11 | Port function control additional expansion register 11 | 16 | 6, 5, 3, 2 | R/W | $0A2C_H$ | $0000_H$ | √ | — | — |
| | | | 15, 9, 6, 5, 3, 2 | | | | — | √ | √ |
| PIBC11 | Port input buffer control register 11 | 16 | 7-0 | R/W | $402C_H$ | $0000_H$ | √ | — | — |
| | | | 15-0 | | | | — | √ | √ |
| PBDC11 | Port bidirection control register 11 | 16 | 7-0 | R/W | $412C_H$ | $0000_H$ | √ | — | — |
| | | | 15-0 | | | | — | √ | √ |
| PIPC11 | Port IP control register 11 | 16 | 7, 6, 3, 2 | R/W | $422C_H$ | $0000_H$ | √ | — | — |
| | | | 10, 9, 7, 6, 3, 2 | | | | — | √ | √ |

Table 2.53 Control Registers (P11) (2/2)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| PU11 | Pull-up option register 11 | 16 | 7-0 | R/W | $432C_H$ | $0000_H$ | √ | — | — |
| | | | 15-0 | | | | — | √ | √ |
| PD11 | Pull-down option register 11 | 16 | 7-0 | R/W | $442C_H$ | $0000_H$ | √ | — | — |
| | | | 15-0 | | | | — | √ | √ |
| PODC11 | Port open drain control register 11 | 32 | 7-0 | R/W | $452C_H$ | $0000\ 0000_H$ | √ | — | — |
| | | | 15-0 | | | | — | √ | √ |
| PDSC11 | Port drive strength control register 11 | 32 | 7-0 | R/W | $462C_H$ | $0000\ 0000_H$ | √ | — | — |
| | | | 15-0 | | | | — | √ | √ |
| PIS11 | Port input buffer selection register 11 | 16 | 7-0 | R/W | $472C_H$ | $FFFF_H$ | √ | — | — |
| | | | 15-0 | | | | — | √ | √ |
| PPROTS11 | Port protection status register 11 | 32 | 0 | R | $4B2C_H$ | $0000\ 0000_H$ | √ | √ | √ |
| PPCMD11 | Port protection command register 11 | 32 | 7-0 | W | $4C2C_H$ | $xxxx\ xx00_H$ | √ | √ | √ |

Note 1. The unused bits are read-only (R). When read, the value after reset is returned.
When writing to unused bits, write the value after reset.

## 2.10.9 Port 12 (P12)

### 2.10.9.1 Alternative Function

Table 2.54 Port 12 (P12)

| Port Mode (PMC12_m =0) | Alternative Mode (PMC12_m =1) | | | | | | | | | | | | | | Special Function | PKG No. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1st Alternative | | 2nd Alternative | | 3rd Alternative | | 4th Alternative | | 5th Alternative | | 6th Alternative | | 7th Alternative | | | | | |
| | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | | 100 pins | 144 pins | 176 pins |
| P12_0 | | CAN2TX | | PWGA56O | TAUB1I10*1 | TAUB1O10*1 | | | | | | | | | | — | 137 | 169 |
| P12_1 | RLIN34RX/ INTP14 | CSIH2CSS5 | | PWGA57O | TAUB1I12*1 | TAUB1O12*1 | | | | | | | RLIN34RX | | | — | 138 | 170 |
| P12_2 | | RLIN34TX | | PWGA58O | TAUB1I14*1 | TAUB1O14*1 | | | | | | | | | | — | 139 | 171 |
| P12_3 | RLIN27RX | | | PWGA68O | | | | | | | | | | | | — | — | 15 |
| P12_4 | | RLIN27TX | | PWGA69O | | | | | | | | | | | | — | — | 16 |
| P12_5 | | PWGA70O | | | | | | | | | | | | | | — | — | 17 |

Note 1. Non-existent in 144 pins PKG.

**CAUTION**

The behavior and performance are not guaranteed when alternative functions are not assigned to the register.

### 2.10.9.2 Control Registers

Table 2.55 Control Registers (P12) (1/2)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| P12 | Port register 12 | 16 | 2-0 | R/W | $0030_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PSR12 | Port set/reset register 12 | 32 | 18-16, 2-0 | R/W | $0130_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 21-16, 5-0 | | | | — | — | √ |
| PPR12 | Port pin read register 12 | 16 | 2-0 | R | $0230_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PM12 | Port mode register 12 | 16 | 2-0 | R/W | $0330_H$ | $FFFF_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PMC12 | Port mode control register 12 | 16 | 2-0 | R/W | $0430_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PFC12 | Port function control register 12 | 16 | 2-0 | R/W | $0530_H$ | $0000_H$ | — | √ | — |
| | | | 4-0 | | | | — | — | √ |
| PFCE12 | Port function control expansion register 12 | 16 | 1 | R/W | $0630_H$ | $0000_H$ | — | √ | — |
| | | | 2-0 | | | | — | — | √ |
| PNOT12 | Port NOT register 12 | 16 | 2-0 | W | $0730_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PMSR12 | Port mode set/reset register 12 | 32 | 18-16, 2-0 | R/W | $0830_H$ | $0000\ FFFF_H$ | — | √ | — |
| | | | 21-16, 5-0 | | | | — | — | √ |
| PMCSR12 | Port mode control set/reset register 12 | 32 | 18-16, 2-0 | R/W | $0930_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 21-16, 5-0 | | | | — | — | √ |
| PFCAE12 | Port function control expansion register 12 | 16 | 1 | R/W | $0A30_H$ | $0000_H$ | — | √ | √ |
| PIBC12 | Port input buffer control register 12 | 16 | 2-0 | R/W | $4030_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PBDC12 | Port bidirection control register 12 | 16 | 2-0 | R/W | $4130_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PU12 | Pull-up option register 12 | 16 | 2-0 | R/W | $4330_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PD12 | Pull-down option register 12 | 16 | 2-0 | R/W | $4430_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |

Table 2.55 Control Registers (P12) (2/2)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| PODC12 | Port open drain control register 12 | 32 | 2-0 | R/W | $4530_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PDSC12 | Port drive strength control register 12 | 32 | 2-0 | R/W | $4630_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PIS12 | Port input buffer selection register 12 | 16 | 2-0 | R/W | $4730_H$ | $FFFF_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PPROTS12 | Port protection status register 12 | 32 | 0 | R | $4B30_H$ | $0000\ 0000_H$ | — | √ | √ |
| PPCMD12 | Port protection command register 12 | 32 | 7-0 | W | $4C30_H$ | $xxxx\ xx00_H$ | — | √ | √ |

Note 1. The unused bits are read-only (R). When read, the value after reset is returned.
When writing to unused bits, write the value after reset.

## 2.10.10 Port 18 (P18)

### 2.10.10.1 Alternative Function

Table 2.56 Port 18 (P18)

| Port Mode (PMC18_m = 0) | Alternative Mode (PMC18_m = 1) | | | | | | | | | | Special Function | PKG No. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1st Alternative | | 2nd Alternative | | 3rd Alternative | | 4th Alternative | | 5th Alternative | | | | | |
| | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | | 100 pins | 144 pins | 176 pins |
| P18_0 | CSIG1RYI | CSIG1RYO | | PWGA61O | | | | | | | ADCA1I0S | — | 115 | 143 |
| P18_1 | | PWGA62O | | | | | | | | | ADCA1I1S | — | 116 | 144 |
| P18_2 | | PWGA63O | | | | | | | | | ADCA1I2S | — | 117 | 145 |
| P18_3 | | PWGA71O*1 | | | | | | | | | ADCA1I3S | — | 118 | 146 |
| P18_4 | | CSIH1CSS4 | | | | | | | | | ADCA1I4S | — | — | 147 |
| P18_5 | | CSIH1CSS5 | | | | | | | | | ADCA1I5S | — | — | 148 |
| P18_6 | | | | | | | | | | | ADCA1I6S | — | — | 149 |
| P18_7 | | | | | | | | | | | ADCA1I7S | — | — | 150 |

Note 1. Non-existent in 144 pins PKG.

**CAUTIONS**

1. The behavior and performance are not guaranteed when alternative functions are not assigned to the register.
2. Use ADC function with initial settings.

### 2.10.10.2 Control Registers

Table 2.57 Control Registers (P18) (1/2)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| P18 | Port register 18 | 16 | 3-0 | R/W | $0048_H$ | $0000_H$ | — | √ | — |
| | | | 7-0 | | | | — | — | √ |
| PSR18 | Port set/reset register 18 | 32 | 19-16, 3-0 | R/W | $0148_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 23-16, 7-0 | | | | — | — | √ |
| PPR18 | Port pin read register 18 | 16 | 3-0 | R | $0248_H$ | $0000_H$ | — | √ | — |
| | | | 7-0 | | | | — | — | √ |
| PM18 | Port mode register 18 | 16 | 3-0 | R/W | $0348_H$ | $FFFF_H$ | — | √ | — |
| | | | 7-0 | | | | — | — | √ |
| PMC18 | Port mode control register 18 | 16 | 3-0 | R/W | $0448_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PFC18 | Port function control register 18 | 16 | 0 | R/W | $0548_H$ | $0000_H$ | — | √ | √ |
| PNOT18 | Port NOT register 18 | 16 | 3-0 | W | $0748_H$ | $0000_H$ | — | √ | — |
| | | | 7-0 | | | | — | — | √ |
| PMSR18 | Port mode set/reset register 18 | 32 | 19-16, 3-0 | R/W | $0848_H$ | $0000\ FFFF_H$ | — | √ | — |
| | | | 23-16, 7-0 | | | | — | — | √ |
| PMCSR18 | Port mode control set/reset register 18 | 32 | 19-16, 3-0 | R/W | $0948_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 21-16, 5-0 | | | | — | — | √ |
| PIBC18 | Port input buffer control register 18 | 16 | 3-0 | R/W | $4048_H$ | $0000_H$ | — | √ | — |
| | | | 7-0 | | | | — | — | √ |
| PBDC18 | Port bidirection control register 18 | 16 | 3-0 | R/W | $4148_H$ | $0000_H$ | — | √ | — |
| | | | 7-0 | | | | — | — | √ |
| PU18 | Pull-up option register 18 | 16 | 3-0 | R/W | $4348_H$ | $0000_H$ | — | √ | — |
| | | | 7-0 | | | | — | — | √ |
| PD18 | Pull-down option register 18 | 16 | 3-0 | R/W | $4448_H$ | $0000_H$ | — | √ | — |
| | | | 7-0 | | | | — | — | √ |
| PODC18 | Port open drain control register 18 | 32 | 3-0 | R/W | $4548_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 7-0 | | | | — | — | √ |
| PDSC18 | Port drive strength control register 18 | 32 | 3-0 | R/W | $4648_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 7-0 | | | | — | — | √ |

Table 2.57 Control Registers (P18) (2/2)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| PIS18 | Port input buffer selection register 18 | 16 | 3-0 | R/W | $4748_H$ | $FFFF_H$ | — | √ | — |
| | | | 7-0 | | | | — | — | √ |
| PPROTS18 | Port protection status register 18 | 32 | 0 | R | $4B48_H$ | $0000\ 0000_H$ | — | √ | √ |
| PPCMD18 | Port protection command register 18 | 32 | 7-0 | W | $4C48_H$ | $xxxx\ xx00_H$ | — | √ | √ |

Note 1. The unused bits are read-only (R). When read, the value after reset is returned.
When writing to unused bits, write the value after reset.

## 2.10.11 Port 20 (P20)

### 2.10.11.1 Alternative Function

Table 2.58 Port 20 (P20)

| Port Mode (PMC20_m =0) | Alternative Mode (PMC20_m = 1) | | | | | | | | | | | | | | Special Function | PKG No. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1st Alternative | | 2nd Alternative | | 3rd Alternative | | 4th Alternative | | 5th Alternative | | 6th Alternative | | 7th Alternative | | | | | |
| | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | | 100 pins | 144 pins | 176 pins |
| P20_0 | RLIN26RX | | | PWGA64O | INTP6 /CAN6RX | | | | | | | | CAN6RX | | | — | — | 118 |
| P20_1 | | RLIN26TX | | PWGA65O | | CAN6TX | | | | | | | | | | — | — | 117 |
| P20_2 | CAN4RX/ INTP4 | | | PWGA66O | RLIN29RX | | | | | | | | CAN4RX | | | — | — | 116 |
| P20_3 | | CAN4TX | | PWGA67O | | RLIN29TX | | | | | | | | | | — | — | 115 |
| P20_4 | RLIN23RX | | | PWGA59O | | | | | | | | | | | | — | 100 | 120 |
| P20_5 | | RLIN23TX | | PWGA60O | | | | | | | | | | | | — | 99 | 119 |

**CAUTION**

The behavior and performance are not guaranteed when alternative functions are not assigned to the register.

### 2.10.11.2 Control Registers

Table 2.59 Control Registers (P20) (1/2)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| P20 | Port register 20 | 16 | 5, 4 | R/W | $0050_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PSR20 | Port set/reset register 20 | 32 | 21-20, 5, 4 | R/W | $0150_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 21-16, 5-0 | | | | — | — | √ |
| PPR20 | Port pin read register 20 | 16 | 5, 4 | R | $0250_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PM20 | Port mode register 20 | 16 | 5, 4 | R/W | $0350_H$ | $FFFF_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PMC20 | Port mode control register 20 | 16 | 5, 4 | R/W | $0450_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PFC20 | Port function control register 20 | 16 | 5, 4 | R/W | $0550_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PFCE20 | Port function control expansion register 20 | 16 | 3-0 | R/W | $0650_H$ | $0000_H$ | — | — | √ |
| PNOT20 | Port NOT register 20 | 16 | 5, 4 | W | $0750_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PMSR20 | Port mode set/reset register 20 | 32 | 21, 20, 5, 4 | R/W | $0850_H$ | $0000\ FFFF_H$ | — | √ | — |
| | | | 21-16, 5-0 | | | | — | — | √ |
| PMCSR20 | Port mode control set/reset register 20 | 32 | 21, 20, 5, 4 | R/W | $0950_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 21-16, 5-0 | | | | — | — | √ |
| PFCAE20 | Port function control additional expansion register 20 | 16 | 2, 0 | R/W | $0A50_H$ | $0000_H$ | — | — | √ |
| PIBC20 | Port input buffer control register 20 | 16 | 5, 4 | R/W | $4050_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PBDC20 | Port bidirection control register 20 | 16 | 5, 4 | R/W | $4150_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PU20 | Pull-up option register 20 | 16 | 5, 4 | R/W | $4350_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PD20 | Pull-down option register 20 | 16 | 5, 4 | R/W | $4450_H$ | $0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |

Table 2.59 Control Registers (P20) (2/2)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| PODC20 | Port open drain control register 20 | 32 | 5, 4 | R/W | $4550_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PDSC20 | Port drive strength control register 20 | 32 | 5, 4 | R/W | $4650_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PIS20 | Port input buffer selection register 20 | 16 | 5, 4 | R/W | $4750_H$ | $FFFF_H$ | — | √ | — |
| | | | 5-0 | | | | — | — | √ |
| PPROTS20 | Port protection status register 20 | 32 | 0 | R | $4B50_H$ | $0000\ 0000_H$ | — | √ | √ |
| PPCMD20 | Port protection command register 20 | 32 | 7-0 | W | $4C50_H$ | $xxxx\ xx00_H$ | — | √ | √ |

Note 1. The unused bits are read-only (R). When read, the value after reset is returned.
When writing to unused bits, write the value after reset.

## 2.10.12 Analog Port 0 (AP0)

### 2.10.12.1 Alternative Function

Table 2.60 Analog Port 0 (AP0)

| Port Mode | Alternative Mode | | | | | | | | | | ADC | PKG No. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1st Alternative | | 2nd Alternative | | 3rd Alternative | | 4th Alternative | | 5th Alternative | | | | | |
| | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | | 100 pins | 144 pins | 176 pins |
| AP0_0 | | | | | | | | | | | ADCA0I0 | 68 | 90 | 106 |
| AP0_1 | | | | | | | | | | | ADCA0I1 | 67 | 89 | 105 |
| AP0_2 | | | | | | | | | | | ADCA0I2 | 66 | 88 | 104 |
| AP0_3 | | | | | | | | | | | ADCA0I3 | 65 | 87 | 103 |
| AP0_4 | | | | | | | | | | | ADCA0I4 | 64 | 86 | 102 |
| AP0_5 | | | | | | | | | | | ADCA0I5 | 63 | 85 | 101 |
| AP0_6 | | | | | | | | | | | ADCA0I6 | 62 | 84 | 100 |
| AP0_7 | | | | | | | | | | | ADCA0I7 | 61 | 83 | 99 |
| AP0_8 | | | | | | | | | | | ADCA0I8 | 60 | 82 | 98 |
| AP0_9 | | | | | | | | | | | ADCA0I9 | 59 | 81 | 97 |
| AP0_10 | | | | | | | | | | | ADCA0I10 | 58 | 80 | 96 |
| AP0_11 | | | | | | | | | | | ADCA0I11 | 57 | 79 | 95 |
| AP0_12 | | | | | | | | | | | ADCA0I12 | 56 | 78 | 94 |
| AP0_13 | | | | | | | | | | | ADCA0I13 | 55 | 77 | 93 |
| AP0_14 | | | | | | | | | | | ADCA0I14 | 54 | 76 | 92 |
| AP0_15 | | | | | | | | | | | ADCA0I15 | 53 | 75 | 91 |

**CAUTION**

Use ADC functions with their initial settings.

### 2.10.12.2 Control Registers

Table 2.61 Control Registers (AP0)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| AP0 | Analog port register 0 | 16 | 15-0 | R/W | $00C8_H$ | $0000_H$ | √ | √ | √ |
| APSR0 | Analog port set/reset register 0 | 32 | 31-16, 15-0 | R/W | $01C8_H$ | $0000\ 0000_H$ | √ | √ | √ |
| APPR0 | Analog port pin read register 0 | 16 | 15-0 | R | $02C8_H$ | $0000_H$ | √ | √ | √ |
| APM0 | Analog port mode register 0 | 16 | 15-0 | R/W | $03C8_H$ | $FFFF_H$ | √ | √ | √ |
| APNOT0 | Analog port NOT register 0 | 16 | 15-0 | W | $07C8_H$ | $0000_H$ | √ | √ | √ |
| APMSR0 | Analog port mode set/reset register 0 | 32 | 31-16, 15-0 | R/W | $08C8_H$ | $0000\ FFFF_H$ | √ | √ | √ |
| APIBC0 | Analog port input buffer control register 0 | 16 | 15-0 | R/W | $40C8_H$ | $0000_H$ | √ | √ | √ |
| APBDC0 | Analog port bidirection control register 0 | 16 | 15-0 | R/W | $41C8_H$ | $0000_H$ | √ | √ | √ |

Note 1. The unused bits are read-only (R). When read, the value after reset is returned.
When writing to unused bits, write the value after reset.

## 2.10.13 Analog Port 1 (AP1)

### 2.10.13.1 Alternative Function

Table 2.62 Analog Port 1 (AP1)

| Port Mode | Alternative Mode | | | | | | | | | | ADC | PKG No. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1st Alternative | | 2nd Alternative | | 3rd Alternative | | 4th Alternative | | 5th Alternative | | | 100 pins | 144 pins | 176 pins |
| | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | | | | |
| AP1_0 | | | | | | | | | | | ADCA1I0 | — | 109 | 133 |
| AP1_1 | | | | | | | | | | | ADCA1I1 | — | 108 | 132 |
| AP1_2 | | | | | | | | | | | ADCA1I2 | — | 107 | 131 |
| AP1_3 | | | | | | | | | | | ADCA1I3 | — | 106 | 130 |
| AP1_4 | | | | | | | | | | | ADCA1I4 | — | 105 | 129 |
| AP1_5 | | | | | | | | | | | ADCA1I5 | — | 104 | 128 |
| AP1_6 | | | | | | | | | | | ADCA1I6 | — | 103 | 127 |
| AP1_7 | | | | | | | | | | | ADCA1I7 | — | 102 | 126 |
| AP1_8 | | | | | | | | | | | ADCA1I8 | — | — | 125 |
| AP1_9 | | | | | | | | | | | ADCA1I9 | — | — | 124 |
| AP1_10 | | | | | | | | | | | ADCA1I10 | — | — | 123 |
| AP1_11 | | | | | | | | | | | ADCA1I11 | — | — | 122 |
| AP1_12 | | | | | | | | | | | ADCA1I12 | — | — | 137 |
| AP1_13 | | | | | | | | | | | ADCA1I13 | — | — | 136 |
| AP1_14 | | | | | | | | | | | ADCA1I14 | — | — | 135 |
| AP1_15 | | | | | | | | | | | ADCA1I15 | — | — | 134 |

**CAUTION**

Use ADC functions with their initial settings.

### 2.10.13.2 Control Registers

Table 2.63 Control Registers (AP1)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| AP1 | Analog port register 1 | 16 | 7-0 | R/W | $00CC_H$ | $0000_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| APSR1 | Analog port set/reset register 1 | 32 | 23-16, 7-0 | R/W | $01CC_H$ | $0000\ 0000_H$ | — | √ | — |
| | | | 31-16, 15-0 | | | | — | — | √ |
| APPR1 | Analog port pin read register 1 | 16 | 7-0 | R | $02CC_H$ | $0000_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| APM1 | Analog port mode register 1 | 16 | 7-0 | R/W | $03CC_H$ | $FFFF_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| APNOT1 | Analog port NOT register 1 | 16 | 7-0 | W | $07CC_H$ | $0000_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| APMSR1 | Analog port mode Set/reset register 1 | 32 | 23-16, 7-0 | R/W | $08CC_H$ | $0000\ FFFF_H$ | — | √ | — |
| | | | 31-16, 15-0 | | | | — | — | √ |
| APIBC1 | Analog port input buffer control register 1 | 16 | 7-0 | R/W | $40CC_H$ | $0000_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |
| APBDC1 | Analog port bidirection control register 1 | 16 | 7-0 | R/W | $41CC_H$ | $0000_H$ | — | √ | — |
| | | | 15-0 | | | | — | — | √ |

Note 1. The unused bits are read-only (R). When read, the value after reset is returned.
When writing to unused bits, write the value after reset.

## 2.10.14 Input Port 0 (IP0)

### 2.10.14.1 Alternative Function

Table 2.64 Input Port 0 (IP0)

| Port Mode | Alternative Mode | | | | | | | | | | Special Function | PKG No. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1st Alternative | | 2nd Alternative | | 3rd Alternative | | 4th Alternative | | 5th Alternative | | | 100 pins | 144 pins | 176 pins |
| | Input | Output | Input | Output | Input | Output | Input | Output | Input | Output | | | | |
| IP0_0 | | | | | | | | | | | XT2 | — | 47 | 57 |

### 2.10.14.2 Control Registers

Table 2.65 Control Registers (IP0)

| Register | Function | Register Size | Effective Bit | | Offset Address | Value after Reset | Device | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Position | R/W*1 | | | 100 pins | 144 pins | 176 pins |
| IPPR0 | Input port pin read register 0 | 16 | 0 | R | $02F0_H$ | $0000_H$ | — | √ | √ |
| IPIBC0 | Port input buffer control register 0 | 16 | 0 | R/W | $40F0_H$ | $0000_H$ | — | √ | √ |

Note 1. The unused bits are read-only (R). When read, the value after reset is returned.
When writing to unused bits, write the value after reset.

**CAUTION**

When the IP0_0/XT2 pin is used as an input port, set the IPIBC0_0 bit to 1 while stopping the SOSC operation. For details on the settings for SOSC operations, see **Section 12.4.2.7, SOSCE — SubOSC Enable Register**. When the IP0_0/XT2 pin is used for the SubOSC (SOSC) not as an input port, set the IPIBC0_0 bit to 0.

# 2.11 Port (Special I/O) Function Overview

This section describes the port (special I/O) functions.

## 2.11.1 Special I/O after Reset

The special port function after reset is deasserted is shown below.

### 2.11.1.1 P8_6: $\overline{\text{RESETOUT}}$

The P8_6 pin ($\overline{\text{RESETOUT}}$ signal) changes PM8.PM8_6 and PODC8.PODC8_6 registers value after reset by OPBT0.RESETOUTEN setting.

The P8_6 pin outputs a low level while a reset is asserted, and pin status of after the reset is different.

(Case 1) : OPBT0.RESETOUTEN = 1

- P8.P8_6 = 0 : Outputs low level
- PM8.PM8_6 = 0 : Output mode
- PODC8.PODC8_6 = 1 : Open-drain

(Case 2) : OPBT0.RESETOUTEN = 0

- P8.P8_6 = 0 : Outputs low level
- PM8.PM8_6 = 1 : Input mode
- PODC8.PODC8_6 = 0 : Push-pull

For detail of OPBT0.RESETOUTEN register, see **Section 37.9.2, OPBT0 — Option Byte 0**, also see **Section 9.4.2, Power-On-Clear (POC) Reset**.

When the P8_6 pin setting is updated with another value, the pin operates by new setting.

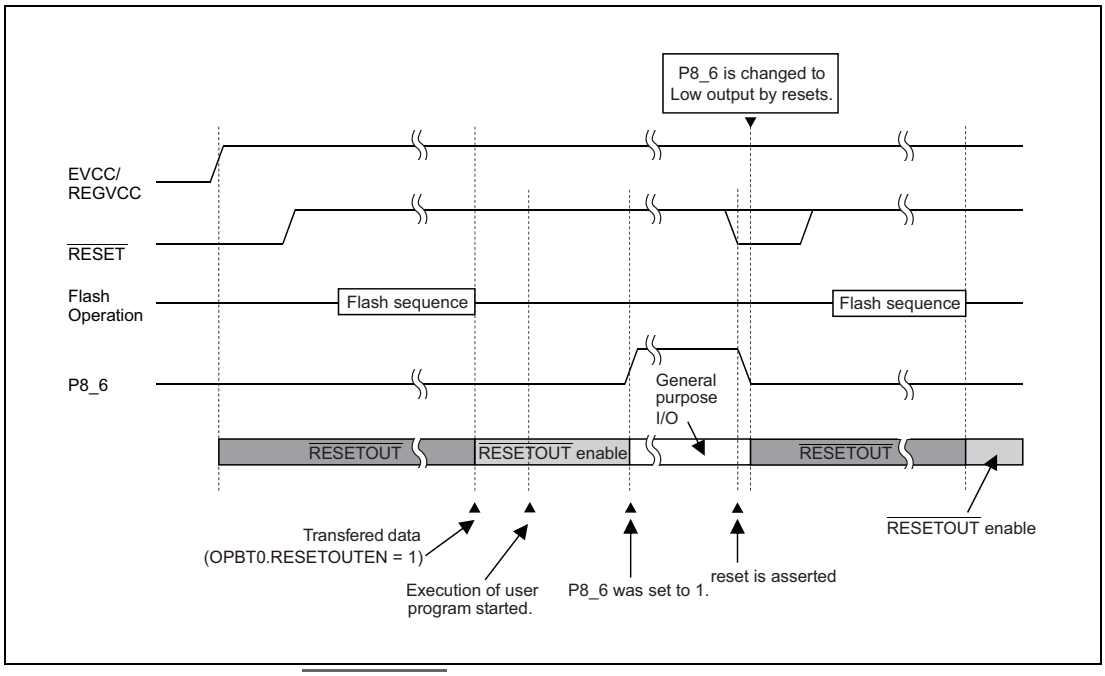



**Figure 2.9 P8_6 Pin ($\overline{\text{RESETOUT}}$ Signal) Operation While a Reset is asserted and released : (Case 1) OPBT0.RESETOUTEN setting is 1**

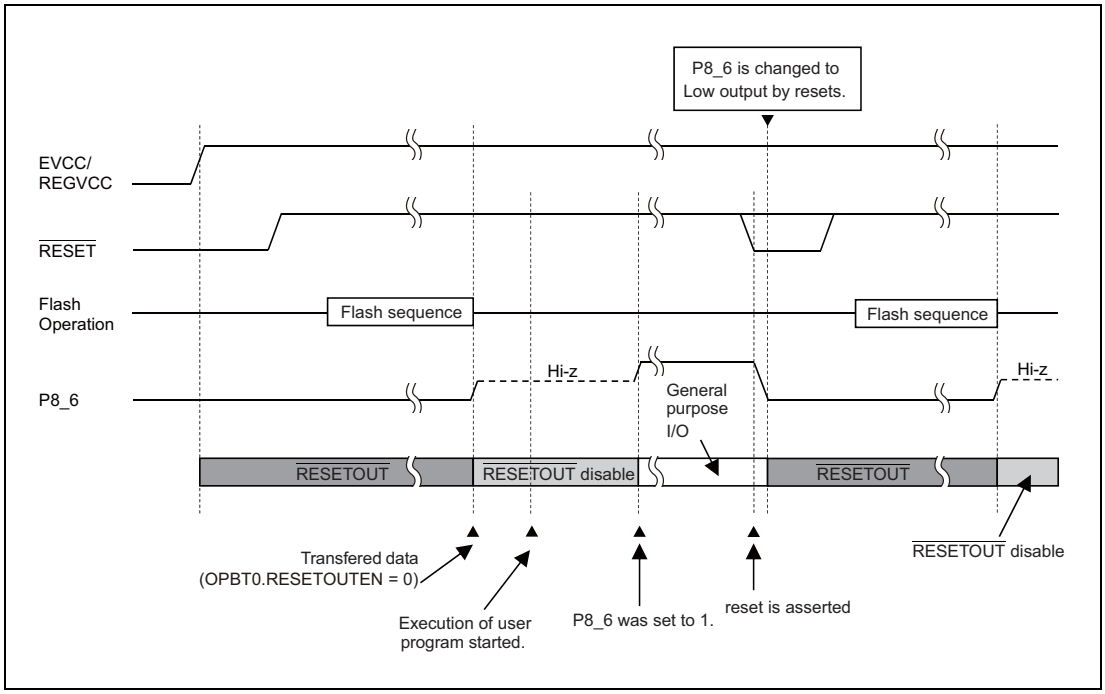



**Figure 2.10 P8_6 Pin ($\overline{\text{RESETOUT}}$ Signal) Operation While a Reset is asserted and released : (Case 2) OPBT0.RESETOUTEN setting is 0**

### 2.11.1.2 JP0_0 to JP0_6: Debug Interface

If the OPJTAG[1:0] setting is the combination below, the pins of the JTAG port group can be used as a debug interface after reset release.

**Table 2.66 Debug Intterface**

| OPJTAG1 | OPJTAG0 | Mode | JP0_0 | JP0_1 | JP0_2 | JP0_3 | JP0_4 | JP0_5 | JP0_6 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Nexus I/F | DCUTDI input | DCUTDO output | DCUTCK input | DCUTMS input | $\overline{\text{DCUTRST}}$ input | $\overline{\text{DCURDY}}$ output | $\overline{\text{EVTO}}$*1 output |
| 0 | 1 | LPD (4-pin) | LPDI input | LPDO output | LPDCLK input | Port/ alternative function | Port/ alternative function | LPDCLK OUT output | Port/ alternative function |
| 1 | 0 | LPD (1-pin) | LPDIO input/ output | Port/ alternative function | Port/ alternative function | Port/ alternative function | Port/ alternative function | Port/ alternative function | Port/ alternative function |

Note 1. Only available for 176 pin and 144 pin devices.

Consequently, the port and alternative functions on these pins cannot be used while the debugger is connected. (The pins with port/alternative function can be used.)

**NOTE**

For the OPJTAG[1:0] settings, see **Section 37.9.2, OPBT0 — Option Byte 0**.

### 2.11.1.3 FPDR(JP0_0), FPDT(JP0_1), FPCK(JP0_2): Flash Programmer

These pins are used for connecting a flash programmer. See *Flash Programmer's Manual* for details.

### 2.11.1.4 Mode Pins

The FLMD0 pin in combination with the P10_8: FLMD1 pin can select serial programming mode.

The FLMD0 pin in combination with the P10_8: FLMD1, the P10_2: MODE1 and the P10_1: MODE0 pins can select boundary scan mode.

The FLMD0 pin in combination with the P10_8: FLMD1, the P10_6: MODE2, the P10_2: MODE1 and the P10_1: MODE0 pins can select user boot mode.

For details on the mode selection, see **Section 6, Operating Mode**.

### 2.11.1.5 IP0_0: XT2

This pin is the SubOSC (SOSC) input pin. When the IPIBC0_0 bit = 1, the IP0_0/XT2 pin is used as an input port. If you make this setting, stop SOSC operation at the same time.

## 2.11.2 A/D Input Alternative I/O

The following ports are permanently connected to A/D input functions. (However, an analog input to the A/D is controlled by the A/D module.)

**Table 2.67 A/D Input Alternative Pins (1/2)**

| Port | A/D Input | Device | | |
|---|---|---|---|---|
| | | 100 pins | 144 pins | 176 pins |
| P8_0 | ADCA0I0S | √ | √ | √ |
| P8_1 | ADCA0I1S | √ | √ | √ |
| P8_2 | ADCA0I4S | √ | √ | √ |
| P8_3 | ADCA0I5S | √ | √ | √ |
| P8_4 | ADCA0I6S | √ | √ | √ |
| P8_5 | ADCA0I7S | √ | √ | √ |
| P8_6 | ADCA0I8S | √ | √ | √ |
| P8_7 | ADCA0I14S | √ | √ | √ |
| P8_8 | ADCA0I15S | √ | √ | √ |
| P8_9 | ADCA0I16S | √ | √ | √ |
| P8_10 | ADCA0I17S | √ | √ | √ |
| P8_11 | ADCA0I18S | √ | √ | √ |
| P8_12 | ADCA0I19S | √ | √ | √ |
| P9_0 | ADCA0I2S | √ | √ | √ |
| P9_1 | ADCA0I3S | √ | √ | √ |
| P9_2 | ADCA0I9S | √ | √ | √ |
| P9_3 | ADCA0I10S | √ | √ | √ |
| P9_4 | ADCA0I11S | √ | √ | √ |
| P9_5 | ADCA0I12S | √ | √ | √ |
| P9_6 | ADCA0I13S | √ | √ | √ |
| P18_0 | ADCA1I0S | — | √ | √ |
| P18_1 | ADCA1I1S | — | √ | √ |
| P18_2 | ADCA1I2S | — | √ | √ |
| P18_3 | ADCA1I3S | — | √ | √ |
| P18_4 | ADCA1I4S | — | — | √ |
| P18_5 | ADCA1I5S | — | — | √ |
| P18_6 | ADCA1I6S | — | — | √ |
| P18_7 | ADCA1I7S | — | — | √ |
| AP0_0 | ADCA0I0 | √ | √ | √ |
| AP0_1 | ADCA0I1 | √ | √ | √ |
| AP0_2 | ADCA0I2 | √ | √ | √ |
| AP0_3 | ADCA0I3 | √ | √ | √ |
| AP0_4 | ADCA0I4 | √ | √ | √ |
| AP0_5 | ADCA0I5 | √ | √ | √ |
| AP0_6 | ADCA0I6 | √ | √ | √ |
| AP0_7 | ADCA0I7 | √ | √ | √ |
| AP0_8 | ADCA0I8 | √ | √ | √ |
| AP0_9 | ADCA0I9 | √ | √ | √ |
| AP0_10 | ADCA0I10 | √ | √ | √ |

Table 2.67 A/D Input Alternative Pins (2/2)

| Port | A/D Input | Device | | |
|---|---|---|---|---|
| | | 100 pins | 144 pins | 176 pins |
| AP0_11 | ADCA0I11 | √ | √ | √ |
| AP0_12 | ADCA0I12 | √ | √ | √ |
| AP0_13 | ADCA0I13 | √ | √ | √ |
| AP0_14 | ADCA0I14 | √ | √ | √ |
| AP0_15 | ADCA0I15 | √ | √ | √ |
| AP1_0 | ADCA1I0 | — | √ | √ |
| AP1_1 | ADCA1I1 | — | √ | √ |
| AP1_2 | ADCA1I2 | — | √ | √ |
| AP1_3 | ADCA1I3 | — | √ | √ |
| AP1_4 | ADCA1I4 | — | √ | √ |
| AP1_5 | ADCA1I5 | — | √ | √ |
| AP1_6 | ADCA1I6 | — | √ | √ |
| AP1_7 | ADCA1I7 | — | √ | √ |
| AP1_8 | ADCA1I8 | — | — | √ |
| AP1_9 | ADCA1I9 | — | — | √ |
| AP1_10 | ADCA1I10 | — | — | √ |
| AP1_11 | ADCA1I11 | — | — | √ |
| AP1_12 | ADCA1I12 | — | — | √ |
| AP1_13 | ADCA1I13 | — | — | √ |
| AP1_14 | ADCA1I14 | — | — | √ |
| AP1_15 | ADCA1I15 | — | — | √ |

## 2.11.3 Special I/O Control

### 2.11.3.1 Direct I/O Control (PIPC)

Some alternative functions take over the input and output control of the ports.

The following table lists all alternative functions where PIPCn.PIPCn_m must be set to 1.

For details, see **Section 2.9.2.3, PIPCn — Port IP Control Register**.

**Table 2.68 Alternative Modes that Require Setting PIPCn.PIPCn_m = 1**

| Function | Alternative functions Name | Port Name | Power Supply Area | Control | Reference Section |
|---|---|---|---|---|---|
| TAPA | TAPA0UP | P10_0 | ISO | U phase Hi-Z control | **Section 29** |
| | TAPA0UN | P10_1 | ISO | | |
| | TAPA0VP | P10_2 | ISO | V phase Hi-Z control | |
| | TAPA0VN | P10_3 | ISO | | |
| | TAPA0WP | P10_4 | ISO | W phase Hi-Z control | |
| | TAPA0WN | P10_5 | ISO | | |
| CSIG | CSIG0SO | P0_13 | AWO | Serial data output control signal | **Section 16** |
| | | P10_6 | ISO | | |
| | CSIG0SC | P0_14 | AWO | Master (1) / slave (0) mode signal | |
| | | P10_7 | ISO | | |
| | CSIG1SO | P11_9 | ISO | Serial data output control signal | |
| | CSIG1SC | P11_10 | ISO | Master (1) / slave (0) mode signal | |
| CSIH | CSIH0SO | P0_3 | AWO | Serial data output control signal | **Section 17** |
| | CSIH0SC | P0_2 | AWO | Master (1) / slave (0) mode signal | |
| | CSIH1SO | P0_5 | AWO | Serial data output control signal | |
| | | P10_2 | ISO | | |
| | CSIH1SC | P0_6 | AWO | Master (1) / slave (0) mode signal | |
| | | P10_1 | ISO | | |
| | CSIH2SO | P11_2 | ISO | Serial data output control signal | |
| | CSIH2SC | P11_3 | ISO | Master (1) / slave (0) mode signal | |
| | CSIH3SO | P11_6 | ISO | Serial data output control signal | |
| | CSIH3SC | P11_7 | ISO | Master (1) / slave (0) mode signal | |

### 2.11.3.2 Input Buffer Control (PISn/JPIS0, JPISA0)

The port input buffer characteristics (Type 1 or Type 2) of this device can be selected using the PISn/JPIS0 register. The applicable pins are shown in the following table.

The JTAG port input buffer characteristics (Type 1/2 or Type 5) of this device can be selected using the JPISA0 register. The applicable pins are shown in **Table 2.70**.

**Table 2.69 Port Input Buffer Characteristics Selection (1/3)**

| Port Name | Input Buffer Selection | | Device | | |
|---|---|---|---|---|---|
| | Type 1 (PISn_m = 0) | Type 2 (PISn_m = 1) | 100 pins | 144 pins | 176 pins |
| P0_0 | SHMT1 | SHMT4 | √ | √ | √ |
| P0_1 | SHMT1 | SHMT4 | √ | √ | √ |
| P0_2 | SHMT1 | SHMT4 | √ | √ | √ |
| P0_3 | SHMT1 | SHMT4 | √ | √ | √ |
| P0_4 | SHMT1 | SHMT4 | √ | √ | √ |
| P0_5 | SHMT1 | SHMT4 | √ | √ | √ |
| P0_6 | SHMT1 | SHMT4 | √ | √ | √ |
| P0_7 | SHMT1 | SHMT4 | √ | √ | √ |
| P0_8 | SHMT1 | SHMT4 | √ | √ | √ |
| P0_9 | SHMT1 | SHMT4 | √ | √ | √ |
| P0_10 | SHMT1 | SHMT4 | √ | √ | √ |
| P0_11 | SHMT1 | SHMT4 | √ | √ | √ |
| P0_12 | SHMT1 | SHMT4 | √ | √ | √ |
| P0_13 | SHMT1 | SHMT4 | √ | √ | √ |
| P0_14 | SHMT1 | SHMT4 | √ | √ | √ |
| P1_0 | SHMT1 | SHMT4 | — | √ | √ |
| P1_2 | SHMT1 | SHMT4 | — | √ | √ |
| P1_3 | SHMT1 | SHMT4 | — | √ | √ |
| P1_4 | SHMT1 | SHMT4 | — | √ | √ |
| P1_5 | SHMT1 | SHMT4 | — | √ | √ |
| P1_6 | SHMT1 | SHMT4 | — | √ | √ |
| P1_7 | SHMT1 | SHMT4 | — | √ | √ |
| P1_8 | SHMT1 | SHMT4 | — | √ | √ |
| P1_9 | SHMT1 | SHMT4 | — | √ | √ |
| P1_10 | SHMT1 | SHMT4 | — | √ | √ |
| P1_11 | SHMT1 | SHMT4 | — | √ | √ |
| P1_12 | SHMT1 | SHMT4 | — | — | √ |
| P1_13 | SHMT1 | SHMT4 | — | — | √ |
| P1_14 | SHMT1 | SHMT4 | — | — | √ |
| P1_15 | SHMT1 | SHMT4 | — | — | √ |
| P2_0 | SHMT1 | SHMT4 | — | — | √ |
| P2_1 | SHMT1 | SHMT4 | — | — | √ |
| P2_2 | SHMT1 | SHMT4 | — | — | √ |
| P2_3 | SHMT1 | SHMT4 | — | — | √ |
| P2_4 | SHMT1 | SHMT4 | — | — | √ |
| P2_5 | SHMT1 | SHMT4 | — | — | √ |
| P2_6 | SHMT1 | SHMT4 | — | — | √ |

Table 2.69 Port Input Buffer Characteristics Selection (2/3)

| Port Name | Input Buffer Selection | | Device | | |
|---|---|---|---|---|---|
| | Type 1 (PISn_m = 0) | Type 2 (PISn_m = 1) | 100 pins | 144 pins | 176 pins |
| P8_0 | SHMT1 | SHMT4 | √ | √ | √ |
| P8_1 | SHMT1 | SHMT4 | √ | √ | √ |
| P8_2 | SHMT1 | SHMT4 | √ | √ | √ |
| P8_3 | SHMT1 | SHMT4 | √ | √ | √ |
| P8_4 | SHMT1 | SHMT4 | √ | √ | √ |
| P8_5 | SHMT1 | SHMT4 | √ | √ | √ |
| P8_6 | SHMT1 | SHMT4 | √ | √ | √ |
| P8_7 | SHMT1 | SHMT4 | √ | √ | √ |
| P8_8 | SHMT1 | SHMT4 | √ | √ | √ |
| P8_9 | SHMT1 | SHMT4 | √ | √ | √ |
| P8_10 | SHMT1 | SHMT4 | √ | √ | √ |
| P8_11 | SHMT1 | SHMT4 | √ | √ | √ |
| P8_12 | SHMT1 | SHMT4 | √ | √ | √ |
| P9_0 | SHMT1 | SHMT4 | √ | √ | √ |
| P9_1 | SHMT1 | SHMT4 | √ | √ | √ |
| P9_2 | SHMT1 | SHMT4 | √ | √ | √ |
| P9_3 | SHMT1 | SHMT4 | √ | √ | √ |
| P9_4 | SHMT1 | SHMT4 | √ | √ | √ |
| P9_5 | SHMT1 | SHMT4 | √ | √ | √ |
| P9_6 | SHMT1 | SHMT4 | √ | √ | √ |
| P10_0 | SHMT1 | SHMT4 | √ | √ | √ |
| P10_1 | SHMT1 | SHMT4 | √ | √ | √ |
| P10_2 | SHMT1 | SHMT4 | √ | √ | √ |
| P10_3 | SHMT1 | SHMT4 | √ | √ | √ |
| P10_4 | SHMT1 | SHMT4 | √ | √ | √ |
| P10_5 | SHMT1 | SHMT4 | √ | √ | √ |
| P10_6 | SHMT1 | SHMT4 | √ | √ | √ |
| P10_7 | SHMT1 | SHMT4 | √ | √ | √ |
| P10_8 | SHMT1 | SHMT4 | √ | √ | √ |
| P10_9 | SHMT1 | SHMT4 | √ | √ | √ |
| P10_10 | SHMT1 | SHMT4 | √ | √ | √ |
| P10_11 | SHMT1 | SHMT4 | √ | √ | √ |
| P10_12 | SHMT1 | SHMT4 | √ | √ | √ |
| P10_13 | SHMT1 | SHMT4 | √ | √ | √ |
| P10_14 | SHMT1 | SHMT4 | √ | √ | √ |
| P10_15 | SHMT1 | SHMT4 | √ | √ | √ |
| P11_0 | SHMT1 | SHMT4 | √ | √ | √ |
| P11_1 | SHMT1 | SHMT4 | √ | √ | √ |
| P11_2 | SHMT1 | SHMT4 | √ | √ | √ |
| P11_3 | SHMT1 | SHMT4 | √ | √ | √ |
| P11_4 | SHMT1 | SHMT4 | √ | √ | √ |
| P11_5 | SHMT1 | SHMT4 | √ | √ | √ |
| P11_6 | SHMT1 | SHMT4 | √ | √ | √ |

Table 2.69 Port Input Buffer Characteristics Selection (3/3)

| Port Name | Input Buffer Selection | | Device | | |
|---|---|---|---|---|---|
| | Type 1 (PISn_m = 0) | Type 2 (PISn_m = 1) | 100 pins | 144 pins | 176 pins |
| P11_7 | SHMT1 | SHMT4 | √ | √ | √ |
| P11_8 | SHMT1 | SHMT4 | — | √ | √ |
| P11_9 | SHMT1 | SHMT4 | — | √ | √ |
| P11_10 | SHMT1 | SHMT4 | — | √ | √ |
| P11_11 | SHMT1 | SHMT4 | — | √ | √ |
| P11_12 | SHMT1 | SHMT4 | — | √ | √ |
| P11_13 | SHMT1 | SHMT4 | — | √ | √ |
| P11_14 | SHMT1 | SHMT4 | — | √ | √ |
| P11_15 | SHMT1 | SHMT4 | — | √ | √ |
| P12_1 | SHMT1 | SHMT4 | — | √ | √ |
| P12_2 | SHMT1 | SHMT4 | — | √ | √ |
| P12_3 | SHMT1 | SHMT4 | — | — | √ |
| P12_4 | SHMT1 | SHMT4 | — | — | √ |
| P12_5 | SHMT1 | SHMT4 | — | — | √ |
| P18_0 | SHMT1 | SHMT4 | — | √ | √ |
| P18_1 | SHMT1 | SHMT4 | — | √ | √ |
| P18_2 | SHMT1 | SHMT4 | — | √ | √ |
| P18_3 | SHMT1 | SHMT4 | — | √ | √ |
| P18_4 | SHMT1 | SHMT4 | — | — | √ |
| P18_5 | SHMT1 | SHMT4 | — | — | √ |
| P18_6 | SHMT1 | SHMT4 | — | — | √ |
| P18_7 | SHMT1 | SHMT4 | — | — | √ |
| P20_0 | SHMT1 | SHMT4 | — | — | √ |
| P20_1 | SHMT1 | SHMT4 | — | — | √ |
| P20_2 | SHMT1 | SHMT4 | — | — | √ |
| P20_3 | SHMT1 | SHMT4 | — | — | √ |
| P20_4 | SHMT1 | SHMT4 | — | √ | √ |
| P20_5 | SHMT1 | SHMT4 | — | √ | √ |

Table 2.70 JTAG port Input Buffer Characteristics Selection

| Port Name | Input Buffer Selection | | | Devices | | |
|---|---|---|---|---|---|---|
| | Type 1 (JPIS0_m=0 & JPISA0_m=0) | Type 2 (JPIS0_m=1 & JPISA0_m=0) | Type 5 (JPISA0_m = 1) | 100 pins | 144 pins | 176 pins |
| JP0_0 | SHMT1 | SHMT4 | TTL*1, *2, *3, *4 | √ | √ | √ |
| JP0_1 | SHMT1 | SHMT4 | — | √ | √ | √ |
| JP0_2 | SHMT1 | SHMT4 | TTL*1, *2, *3 | √ | √ | √ |
| JP0_3 | SHMT1 | SHMT4 | TTL*1, *2 | √ | √ | √ |
| JP0_4 | — | SHMT4 | —*1, *2 | √ | √ | √ |
| JP0_5 | SHMT1 | SHMT4 | — | √ | √ | √ |
| JP0_6 | SHMT1 | SHMT4 | — | — | √ | √ |

Note 1. TTL is selected for Boundary scan mode without JPISA0 register setting.
Note 2. TTL is selected for Nexus in normal operating mode without JPISA0 register setting.
Note 3. TTL is selected for LPD (4-pin) in normal operating mode without JPISA0 register setting.
Note 4. TTL is selected for LPD (1-pin) in normal operating mode without JPISA0 register setting.

**NOTES**

1. For the SHMT1, SHMT4, and TTL pin characteristics, see the **Section 40, Electrical Characteristics**.
2. For the input buffer after reset, type 2 (SHMT4) is selected.

### 2.11.3.3 Output Buffer Control (PDSC)

The port output driver strength (slow mode/fast mode) can be selected using the PDSCn register. The applicable pins are shown in the following table. Only slow mode is supported for ports other than those listed below.

**Table 2.71 Output Buffer Characteristics Selection (1/3)**

| Port Name | Output Drive Strength Selection | | Device | | |
|---|---|---|---|---|---|
| | Slow Mode (PDSCn_m = 0) | Fast Mode (PDSCn_m = 1) | 100 pins | 144 pins | 176 pins |
| JP0_1 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| JP0_2 | 10 MHz | 40 MHz | √ | √ | √ |
| JP0_3 | 10 MHz | 40 MHz | √ | √ | √ |
| JP0_5 | 10 MHz | 40 MHz | √ | √ | √ |
| JP0_6 | 10 MHz | 40 MHz | — | √ | √ |
| P0_0 | 10 MHz | 40 MHz | √ | √ | √ |
| P0_1 | 10 MHz | 40 MHz | √ | √ | √ |
| P0_2 | 10 MHz | 40 MHz*1 | √ | √ | √ |
| P0_3 | 10 MHz | 40 MHz*1 | √ | √ | √ |
| P0_4 | 10 MHz | 40 MHz | √ | √ | √ |
| P0_5 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P0_6 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P0_7 | 10 MHz | 40 MHz | √ | √ | √ |
| P0_8 | 10 MHz | 40 MHz | √ | √ | √ |
| P0_9 | 10 MHz | 40 MHz | √ | √ | √ |
| P0_10 | 10 MHz | 40 MHz | √ | √ | √ |
| P0_11 | 10 MHz | 40 MHz | √ | √ | √ |
| P0_12 | 10 MHz | 40 MHz | √ | √ | √ |
| P0_13 | 10 MHz | 40 MHz | √ | √ | √ |
| P0_14 | 10 MHz | 40 MHz | √ | √ | √ |
| P1_0 | 10 MHz | 40 MHz | — | √ | √ |
| P1_1 | 10 MHz | 40 MHz | — | √ | √ |
| P1_2 | 10 MHz | 40 MHz | — | √ | √ |
| P1_3 | 10 MHz | 40 MHz | — | √ | √ |
| P1_4 | 10 MHz | 40 MHz | — | √ | √ |
| P1_5 | 10 MHz | 40 MHz | — | √ | √ |
| P1_6 | 10 MHz | 40 MHz | — | √ | √ |
| P1_7 | 10 MHz | 40 MHz | — | √ | √ |
| P1_8 | 10 MHz | 40 MHz | — | √ | √ |
| P1_9 | 10 MHz | 40 MHz | — | √ | √ |
| P1_10 | 10 MHz | 40 MHz | — | √ | √ |
| P1_11 | 10 MHz | 40 MHz | — | √ | √ |
| P1_12 | 10 MHz | 40 MHz | — | — | √ |
| P1_13 | 10 MHz | 40 MHz | — | — | √ |
| P1_14 | 10 MHz | 40 MHz | — | — | √ |
| P1_15 | 10 MHz | 40 MHz | — | — | √ |
| P2_0 | 10 MHz | 40 MHz | — | — | √ |
| P2_1 | 10 MHz | 40 MHz | — | — | √ |

Table 2.71 Output Buffer Characteristics Selection (2/3)

| Port Name | Output Drive Strength Selection | | Device | | |
|---|---|---|---|---|---|
| | Slow Mode (PDSCn_m = 0) | Fast Mode (PDSCn_m = 1) | 100 pins | 144 pins | 176 pins |
| P2_2 | 10 MHz | 40 MHz | — | — | √ |
| P2_3 | 10 MHz | 40 MHz | — | — | √ |
| P2_4 | 10 MHz | 40 MHz | — | — | √ |
| P2_5 | 10 MHz | 40 MHz | — | — | √ |
| P2_6 | 10 MHz | 40 MHz | — | — | √ |
| P10_0 | 10 MHz | 40 MHz | √ | √ | √ |
| P10_1 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P10_2 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P10_3 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P10_4 | 10 MHz | 40 MHz | √ | √ | √ |
| P10_5 | 10 MHz | 40 MHz | √ | √ | √ |
| P10_6 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P10_7 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P10_8 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P10_9 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P10_10 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P10_11 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P10_12 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P10_13 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P10_14 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P10_15 | 10 MHz | 40 MHz | √ | √ | √ |
| P11_0 | 10 MHz | 40 MHz | √ | √ | √ |
| P11_1 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P11_2 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P11_3 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P11_4 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P11_5 | 10 MHz | 40 MHz | √ | √ | √ |
| P11_6 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P11_7 | 10 MHz | 40 MHz*2 | √ | √ | √ |
| P11_8 | 10 MHz | 40 MHz | — | √ | √ |
| P11_9 | 10 MHz | 40 MHz | — | √ | √ |
| P11_10 | 10 MHz | 40 MHz | — | √ | √ |
| P11_11 | 10 MHz | 40 MHz | — | √ | √ |
| P11_12 | 10 MHz | 40 MHz | — | √ | √ |
| P11_13 | 10 MHz | 40 MHz | — | √ | √ |
| P11_14 | 10 MHz | 40 MHz | — | √ | √ |
| P11_15 | 10 MHz | 40 MHz | — | √ | √ |
| P12_0 | 10 MHz | 40 MHz*2 | — | √ | √ |
| P12_1 | 10 MHz | 40 MHz*2 | — | √ | √ |
| P12_2 | 10 MHz | 40 MHz*2 | — | √ | √ |
| P12_3 | 10 MHz | 40 MHz | — | — | √ |
| P12_5 | 10 MHz | 40 MHz | — | — | √ |

Table 2.71 Output Buffer Characteristics Selection (3/3)

| Port Name | Output Drive Strength Selection | | Device | | |
|---|---|---|---|---|---|
| | Slow Mode (PDSCn_m = 0) | Fast Mode (PDSCn_m = 1) | 100 pins | 144 pins | 176 pins |
| P18_0 | 10 MHz | 40 MHz*2 | — | √ | √ |
| P18_1 | 10 MHz | 40 MHz*2 | — | √ | √ |
| P18_2 | 10 MHz | 40 MHz*2 | — | √ | √ |
| P18_3 | 10 MHz | 40 MHz*2 | — | √ | √ |
| P18_4 | 10 MHz | 40 MHz*2 | — | — | √ |
| P18_5 | 10 MHz | 40 MHz*2 | — | — | √ |
| P18_6 | 10 MHz | 40 MHz*2 | — | — | √ |
| P18_7 | 10 MHz | 40 MHz*2 | — | — | √ |
| P20_0 | 10 MHz | 40 MHz | — | — | √ |
| P20_1 | 10 MHz | 40 MHz | — | — | √ |
| P20_2 | 10 MHz | 40 MHz | — | — | √ |
| P20_3 | 10 MHz | 40 MHz | — | — | √ |
| P20_4 | 10 MHz | 40 MHz | — | √ | √ |
| P20_5 | 10 MHz | 40 MHz | — | √ | √ |

Note 1. Supports Cload: 100pF

Note 2. Supports Cload: 50pF (Set fast mode if the load capacitance of CSIH is 50 pF.)

## 2.12 Noise Filter & Edge/Level Detector

The input signals at some pins are passed through a filter to remove noise and glitches. The RH850/F1K supports both analog and digital filters.

It also supports the function for edge and level detection after the signals have passed through a filter.

The first part of this section provides an overview of port input pins that are equipped with a filter and the filter type, noise filter & edge/level detection control registers and control bits, and register addresses.

For details on the digital/analog filter function and noise filter & edge/level detection control registers, see **Section 2.13, Description of Port Noise Filter & Edge/Level Detection**.

**NOTE**

In this section, <name> in the noise filter control register represents the peripheral function connected to a filter.

### 2.12.1 Port Filter Assignment

A list of the input pins that incorporate an analog or digital filter is provided below.

#### 2.12.1.1 Input Pins that Incorporate Analog Filter Type A

The input pins of analog filter type A incorporate an analog filter and edge/level detection function. Edge/level detection is controlled by the following registers.

- Filter control register FCLA0CTLm_<name> (m = 0 to 7)
  A dedicated FCLA0CTLm_<name> register is provided for each pin in a port that incorporates an analog filter.

**Table 2.72 Input Pins that Incorporate Analog Filter Type A**

| Module Name | Input Pin | FCLA0CTL Register Configuration | | Device | | |
|---|---|---|---|---|---|---|
| | | Register | Address | 100 pins | 144 pins | 176 pins |
| FCLA0 | NMI | FCLA0CTL0_NMI | FFC3 4000$_H$ | √ | √ | √ |
| | INTP0 | FCLA0CTL0_INTPL | FFC3 4020$_H$ | √ | √ | √ |
| | INTP1 | FCLA0CTL1_INTPL | FFC3 4024$_H$ | √ | √ | √ |
| | INTP2 | FCLA0CTL2_INTPL | FFC3 4028$_H$ | √ | √ | √ |
| | INTP3 | FCLA0CTL3_INTPL | FFC3 402C$_H$ | √ | √ | √ |
| | INTP4 | FCLA0CTL4_INTPL | FFC3 4030$_H$ | √ | √ | √ |
| | INTP5 | FCLA0CTL5_INTPL | FFC3 4034$_H$ | √ | √ | √ |
| | INTP6 | FCLA0CTL6_INTPL | FFC3 4038$_H$ | √ | √ | √ |
| | INTP7 | FCLA0CTL7_INTPL | FFC3 403C$_H$ | √ | √ | √ |
| | INTP8 | FCLA0CTL0_INTPH | FFC3 4040$_H$ | √ | √ | √ |
| | INTP9 | FCLA0CTL1_INTPH | FFC3 4044$_H$ | — | √ | √ |
| | INTP10 | FCLA0CTL2_INTPH | FFC3 4048$_H$ | √ | √ | √ |
| | INTP11 | FCLA0CTL3_INTPH | FFC3 404C$_H$ | √ | √ | √ |
| | INTP12 | FCLA0CTL4_INTPH | FFC3 4050$_H$ | √ | √ | √ |
| | INTP13 | FCLA0CTL5_INTPH | FFC3 4054$_H$ | √ | √ | √ |
| | INTP14 | FCLA0CTL6_INTPH | FFC3 4058$_H$ | — | √ | √ |
| | INTP15 | FCLA0CTL7_INTPH | FFC3 405C$_H$ | — | √ | √ |

### 2.12.1.2 Input Pins that Incorporate Analog Filter Type B

The input pins of analog filter type B incorporate an analog filter. Edge/level detection is controlled by the registers for individual peripheral functions.

**Table 2.73 Input Pins that Incorporate Analog Filter Type B**

| Input Pin | Edge/Level Detection | Device | | |
|---|---|---|---|---|
| | | 100 pins | 144 pins | 176 pins |
| TAUJ0I0 | Edge detection*1 | √ | √ | √ |
| TAUJ0I1 | Edge detection*1 | √ | √ | √ |
| TAUJ0I2 | Edge detection*1 | √ | √ | √ |
| TAUJ0I3 | Edge detection*1 | √ | √ | √ |
| TAUJ1I0 | Edge detection*1 | √ | √ | √ |
| TAUJ1I1 | Edge detection*1 | √ | √ | √ |
| TAUJ1I2 | Edge detection*1 | √ | √ | √ |
| TAUJ1I3 | Edge detection*1 | √ | √ | √ |
| TAPA0ESO | Edge detection*2 | √ | √ | √ |
| KR0I0 | Low level detection | √ | √ | √ |
| KR0I1 | Low level detection | √ | √ | √ |
| KR0I2 | Low level detection | √ | √ | √ |
| KR0I3 | Low level detection | √ | √ | √ |
| KR0I4 | Low level detection | √ | √ | √ |
| KR0I5 | Low level detection | √ | √ | √ |
| KR0I6 | Low level detection | √ | √ | √ |
| KR0I7 | Low level detection | √ | √ | √ |

Note 1. For details on edge detection for TAUJ, see **Section 26.3.3.4, TAUJnCMURm — TAUJn Channel Mode User Register**.

Note 2. For details on edge detection for TAPA, see **Section 29.3.2, TAPAnCTL0 — TAPA Control Register 0**.

### 2.12.1.3 Input Pins that Incorporate Analog Filter Type C

The input pins of analog filter type C only incorporate an analog filter function.

**Table 2.74 Input Pins that Incorporate Analog Filter Type C**

| Input Pin |
|---|
| FLMD0 |
| FLMD1 |
| MODE0 |
| MODE1 |
| MODE2 |
| $\overline{\text{RESET}}$ |
| $\overline{\text{DCUTRST}}$ |

#### 2.12.1.4 Input Pins that Incorporate Digital Filter Type D

The input pins of digital filter type D incorporate a digital filter and edge detection function. The digital filter and edge detection are controlled by the following registers.

- Filter control register FCLA0CTLm_<name> (m = 0 to 2)
  Each port with a digital filter has a special FCLA0CTLm_<name> register.
- Digital noise elimination control register DNFA<name>CTL
  Each DNFA<name>CTL control register controls digital filter processing for three input signals per group.
- Digital noise elimination enable register DNFA<name>EN
  The setting of the DNFA<name>ENL[2:0] bits in DNFA<name>EN enables or disables digital noise elimination for three input signals per group.

Table 2.75 Input Pins that Incorporate Digital Filter Type D

| Input Pin | Device | | | Digital Noise Elimination Control Register | | Digital Noise Elimination Enable Register | | | Filter Control Register | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 100 pin | 144 pin | 176 pin | Control Register | Address | Control Register | Control Bit | Address | Control Register | Address |
| ADCA0TRG0 | √ | √ | √ | DNFAADCTL0CTL | FFC3 00A0$_H$ | DNFAADCTL0EN (DNFAADCTL0ENL) | DNFAADCTL0ENL0 | FFC3 00A4$_H$ (FFC3 00AC$_H$) | FCLA0CTL0 _ADC0 | FFC3 4060$_H$ |
| ADCA0TRG1 | √ | √ | √ | | | | DNFAADCTL0ENL1 | | FCLA0CTL1 _ADC0 | FFC3 4064$_H$ |
| ADCA0TRG2 | √ | √ | √ | | | | DNFAADCTL0ENL2 | | FCLA0CTL2 _ADC0 | FFC3 4068$_H$ |
| ADCA1TRG0 | — | √ | √ | DNFAADCTL1CTL | FFC3 00C0$_H$ | DNFAADCTL1EN (DNFAADCTL1ENL) | DNFAADCTL1ENL0 | FFC3 00C4$_H$ (FFC3 00CC$_H$) | FCLA0CTL0 _ADC1 | FFC3 4080$_H$ |
| ADCA1TRG1 | — | √ | √ | | | | DNFAADCTL1ENL1 | | FCLA0CTL1 _ADC1 | FFC3 4084$_H$ |
| ADCA1TRG2 | — | √ | √ | | | | DNFAADCTL1ENL2 | | FCLA0CTL2 _ADC1 | FFC3 4088$_H$ |

#### 2.12.1.5 Input Pins that Incorporate Digital Filter Type E

The input pins of digital filter type E incorporate a digital filter. The digital filter is controlled by the following registers. Edge detection is controlled by the registers for individual peripheral functions.

- Digital noise elimination control register DNFA<name>CTL
  Each DNFA<name>CTL control register controls digital filter processing for up to 16 input signals per group.
- Digital noise elimination enable register DNFA<name>EN
  The setting of the DNFA<name>ENL[7:0] and DNFA<name>ENH[7:0] bits in DNFA<name>EN enables or disables digital noise elimination for up to 16 input signals per group.

Table 2.76 Input Pins that Incorporate Digital Filter Type E (1/3)

| Input Pin | Devises | | | Digital Noise Elimination Control Register | | Digital Noise Elimination Enable Register | | | Edge Detection |
|---|---|---|---|---|---|---|---|---|---|
| | 100 pin | 144 pin | 176 pin | Control Register | Address | Control Register | Control Bit | Address | Register Name |
| TAUD0I0 | √ | √ | √ | DNFATAUD0ICTL | FFC3 0000$_H$ | DNFATAUD0IEN (DNFATAUD0IENH/ DNFATAUD0IENL) | DNFATAUD0IENL0 | FFC3 0004$_H$ (FFC3 0008$_H$/ FFC3 000C$_H$) | *1 |
| TAUD0I1 | √ | √ | √ | | | | DNFATAUD0IENL1 | | |
| TAUD0I2 | √ | √ | √ | | | | DNFATAUD0IENL2 | | |
| TAUD0I3 | √ | √ | √ | | | | DNFATAUD0IENL3 | | |
| TAUD0I4 | √ | √ | √ | | | | DNFATAUD0IENL4 | | |
| TAUD0I5 | √ | √ | √ | | | | DNFATAUD0IENL5 | | |
| TAUD0I6 | √ | √ | √ | | | | DNFATAUD0IENL6 | | |
| TAUD0I7 | √ | √ | √ | | | | DNFATAUD0IENL7 | | |
| TAUD0I8 | √ | √ | √ | | | | DNFATAUD0IENH0 | | |
| TAUD0I9 | √ | √ | √ | | | | DNFATAUD0IENH1 | | |
| TAUD0I10 | √ | √ | √ | | | | DNFATAUD0IENH2 | | |
| TAUD0I11 | √ | √ | √ | | | | DNFATAUD0IENH3 | | |
| TAUD0I12 | √ | √ | √ | | | | DNFATAUD0IENH4 | | |
| TAUD0I13 | √ | √ | √ | | | | DNFATAUD0IENH5 | | |
| TAUD0I14 | √ | √ | √ | | | | DNFATAUD0IENH6 | | |
| TAUD0I15 | √ | √ | √ | | | | DNFATAUD0IENH7 | | |

Table 2.76 Input Pins that Incorporate Digital Filter Type E (2/3)

| Input Pin | Devises | | | Digital Noise Elimination Control Register | | Digital Noise Elimination Enable Register | | | Edge Detection |
|---|---|---|---|---|---|---|---|---|---|
| | 100 pin | 144 pin | 176 pin | Control Register | Address | Control Register | Control Bit | Address | Register Name |
| TAUB0I0 | √ | √ | √ | DNFATAUB0ICTL | FFC3 0020$_H$ | DNFATAUB0IEN (DNFATAUB0IENH/ DNFATAUB0IENL) | DNFATAUB0IENL0 | FFC3 0024$_H$ (FFC3 0028$_H$/ FFC3 002C$_H$) | *2 |
| TAUB0I1 | √ | √ | √ | | | | DNFATAUB0IENL1 | | |
| TAUB0I2 | √ | √ | √ | | | | DNFATAUB0IENL2 | | |
| TAUB0I3 | √ | √ | √ | | | | DNFATAUB0IENL3 | | |
| TAUB0I4 | √ | √ | √ | | | | DNFATAUB0IENL4 | | |
| TAUB0I5 | √ | √ | √ | | | | DNFATAUB0IENL5 | | |
| TAUB0I6 | √ | √ | √ | | | | DNFATAUB0IENL6 | | |
| TAUB0I7 | √ | √ | √ | | | | DNFATAUB0IENL7 | | |
| TAUB0I8 | √ | √ | √ | | | | DNFATAUB0IENH0 | | |
| TAUB0I9 | √ | √ | √ | | | | DNFATAUB0IENH1 | | |
| TAUB0I10 | √ | √ | √ | | | | DNFATAUB0IENH2 | | |
| TAUB0I11 | √ | √ | √ | | | | DNFATAUB0IENH3 | | |
| TAUB0I12 | √ | √ | √ | | | | DNFATAUB0IENH4 | | |
| TAUB0I13 | √ | √ | √ | | | | DNFATAUB0IENH5 | | |
| TAUB0I14 | √ | √ | √ | | | | DNFATAUB0IENH6 | | |
| TAUB0I15 | √ | √ | √ | | | | DNFATAUB0IENH7 | | |

Table 2.76 Input Pins that Incorporate Digital Filter Type E (3/3)

| Input Pin | Devises | | | Digital Noise Elimination Control Register | | Digital Noise Elimination Enable Register | | | Edge Detection |
|---|---|---|---|---|---|---|---|---|---|
| | 100 pin | 144 pin | 176 pin | Control Register | Address | Control Register | Control Bit | Address | Register Name |
| TAUB1I0 | — | — | √ | DNFATAUB1ICTL | FFC3 0040$_H$ | DNFATAUB1IEN (DNFATAUB1IENH/ DNFATAUB1IENL) | DNFATAUB1IENL0 | FFC3 0044$_H$ (FFC3 0048$_H$/ FFC3 004C$_H$) | *2 |
| TAUB1I1 | — | — | √ | | | | DNFATAUB1IENL1 | | |
| TAUB1I2 | — | — | √ | | | | DNFATAUB1IENL2 | | |
| TAUB1I3 | — | — | √ | | | | DNFATAUB1IENL3 | | |
| TAUB1I4 | — | — | √ | | | | DNFATAUB1IENL4 | | |
| TAUB1I5 | — | — | √ | | | | DNFATAUB1IENL5 | | |
| TAUB1I6 | — | — | √ | | | | DNFATAUB1IENL6 | | |
| TAUB1I7 | — | — | √ | | | | DNFATAUB1IENL7 | | |
| TAUB1I8 | — | — | √ | | | | DNFATAUB1IENH0 | | |
| TAUB1I9 | — | — | √ | | | | DNFATAUB1IENH1 | | |
| TAUB1I10 | — | — | √ | | | | DNFATAUB1IENH2 | | |
| TAUB1I11 | — | — | √ | | | | DNFATAUB1IENH3 | | |
| TAUB1I12 | — | — | √ | | | | DNFATAUB1IENH4 | | |
| TAUB1I13 | — | — | √ | | | | DNFATAUB1IENH5 | | |
| TAUB1I14 | — | — | √ | | | | DNFATAUB1IENH6 | | |
| TAUB1I15 | — | — | √ | | | | DNFATAUB1IENH7 | | |
| ENCA0TIN0 | √ | √ | √ | DNFAENCA0ICTL | FFC3 0060$_H$ | DNFAENCA0IEN (DNFAENCA0IENL) | DNFAENCA0IENL0 | FFC3 0064$_H$ (FFC3 006C$_H$) | *3 |
| ENCA0TIN1 | √ | √ | √ | | | | DNFAENCA0IENL1 | | |
| ENCA0E0 | √ | √ | √ | | | | DNFAENCA0IENL2 | | |
| ENCA0E1 | √ | √ | √ | | | | DNFAENCA0IENL3 | | |
| ENCA0EC | √ | √ | √ | | | | DNFAENCA0IENL4 | | |

Note 1. For the setting for TAUD edge detection, see **Section 25.3.3.4, TAUDnCMURm — TAUDn Channel Mode User Register**.

Note 2. For the setting for TAUB edge detection, see **Section 24.3.3.4, TAUBnCMURm — TAUBn Channel Mode User Register**

Note 3. For the setting for ENCA edge detection, see **Section 28.3.3, ENCAnIOC0 — ENCAn I/O Control Register 0**.

## 2.12.2 Clock Supply for Port Filters

The following table shows the clock supply for each filter type in each port domain.

**Table 2.77 Clock Supply for Port Filters**

| Peripheral Function | Port Domain*1 | Filter Type | Filter Clock | Setting Register | |
|---|---|---|---|---|---|
| | | | | Source Clock Selection | Clock Selection |
| ADCA0 | Always-On area | Digital filter type D | DNFATCKI | CKSC_AADCAS_CTL | CKSC_AADCAD_CTL |
| ADCA1 | Isolated area | Digital filter type D | DNFATCKI | CKSC_IADCAS_CTL | CKSC_IADCAD_CTL |
| TAUD0 | Isolated area | Digital filter type E | DNFATCKI | CKSC_IPERI1S_CTL | — |
| TAUB0 | Isolated area | Digital filter type E | DNFATCKI | CKSC_IPERI2S_CTL | — |
| TAUB1 | Isolated area | Digital filter type E | DNFATCKI | CKSC_IPERI2S_CTL | — |
| ENCA0 | Isolated area | Digital filter type E | DNFATCKI | CKSC_IPERI1S_CTL | — |

Note 1. Power Domain

**NOTE**

For the Setting Register, see **Section 12.4.3, Clock Selector Control Register**.

# 2.13 Description of Port Noise Filter & Edge/Level Detection

External signals pass through different types of filters according to the use of each external input signal.

**NOTE**

In this section, <name> in the noise filter control register represents the peripheral function connected to a filter.

## 2.13.1 Overview

### 2.13.1.1 Analog Filter Types

Analog filters have fixed characteristics.

- Type A: An analog filter with edge detection or level detection.
  Used for external interrupt signals.
- Type B: An analog filter
  Edge detection is performed by each peripheral function. Used for the timer input signals, asynchronous Hi-Z control input signals, and key return input signals.
- Type C: An analog filter only
  Used for the external $\overline{\text{RESET}}$ input and mode signals.

### 2.13.1.2 Digital Filter Types

The digital filter characteristics can be adjusted to suit the application.

- Type D: A digital filter with edge detection.
  Used for the A/D converter external trigger pin.
- Type E: A digital filter. Edge detection is performed by each peripheral function.
  Used for the timer input signals and encoder input signals.

## 2.13.2 Analog Filters

### 2.13.2.1 Analog Filter Characteristic

See the **Section 40, Electrical Characteristics** for the input conditions for signals input to pins that incorporate an analog filter.

### 2.13.2.2 Analog Filter Control Registers

A dedicated FCLA0CTLm_<name> register or control register in the peripheral macro is provided for input pins that incorporate an analog filter.

The assignment of the input signals to the control registers and their addresses are given in **Table 2.72, Input Pins that Incorporate Analog Filter Type A**, in **Section 2.12.1, Port Filter Assignment**.

### 2.13.2.3 Analog Filter in Standby Mode

Analog filters for the function of waking-up from the DeepSTOP mode are located in the Always-On area (AWO area). Analog filters in the Always-On area (AWO area) always operate.

The analog filter in standby mode and its wake-up capability depend on the filter types. See the description of the analog filter types below.

#### (1) Analog Filter Type A

A block diagram of analog filter type A is shown below.

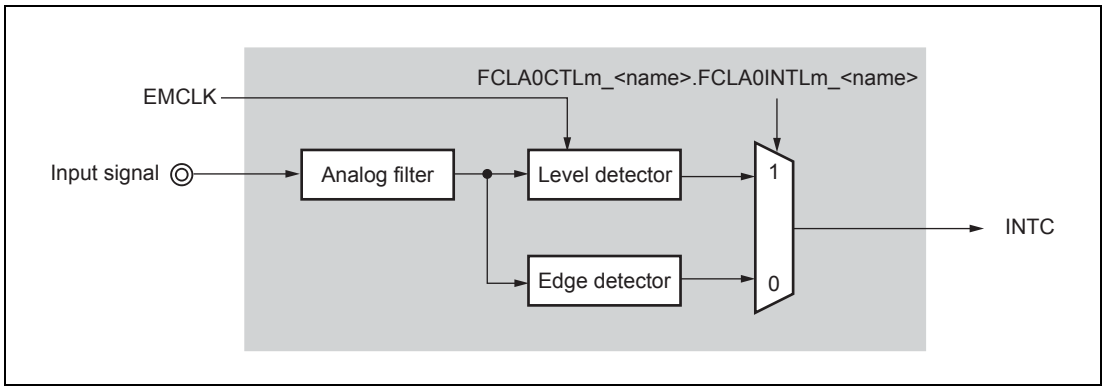



**Figure 2.11 Block Diagram of Analog Filter Type A**

After passing an external signal through the filter to eliminate noise and glitches, an output signal is generated according to whether an event is detected; that is whether a specified level is detected or whether a change in the level (an edge) occurs.

Whether a level or an edge is detected is selected by the control bit FCLA0CTLm_<name>.FCLA0INTLm_<name>.

- FCLA0INTLm_<name> bit = 0: Edge detection

  Whether a rising or falling edge is detected can be specified by setting the FCLA0CTLm_<name>.FCLA0INTRm_<name> and FCLA0CTLm_<name>.FCLA0INTFm_<name> bits.

- FCLA0INTLm_<name> bit = 1: Level detection

  The detection of a high level or low level can be specified by setting FCLA0CTLm_<name>.FCLA0INTRm_<name> bit.

The table below summarizes the detection conditions of the analog filter.

**Table 2.78 Analog Filter Event Detection Conditions**

| FCLA0INTLm_<name> | FCLA0INTFm_<name> | FCLA0INTRm_<name> | Edge Detection | Level Detection |
|---|---|---|---|---|
| 0 | 0 | 0 | No edge detected | Disabled |
| | 0 | 1 | Rising edge | |
| | 1 | 0 | Falling edge | |
| | 1 | 1 | Rising and falling edges | |
| 1 | X | 0 | Disabled | Low level |
| | X | 1 | | High level |

#### Analog filter type A in Standby mode

The output signal of an analog filter type A can always be used as a standby mode wake-up signal.

### (2) Analog filter type B

A block diagram of analog filter type B is shown below.

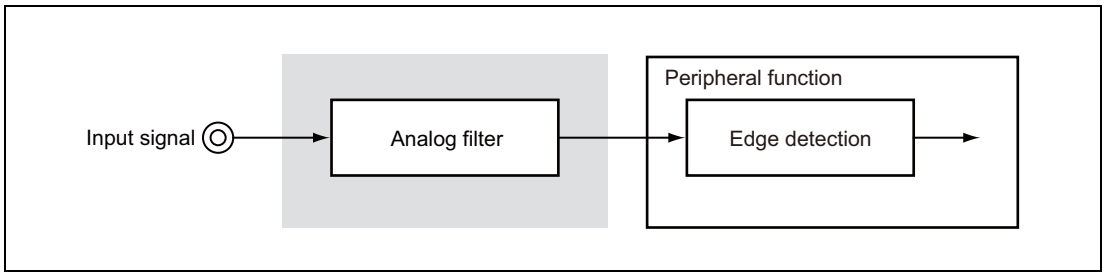



**Figure 2.12 Block Diagram of Analog Filter Type B**

#### Analog filter type B in Standby mode

The output signal of an analog filter type B can always be used as a standby mode wake-up signal.

### (3) Analog filter type C

A block diagram of analog filter type C is shown below.

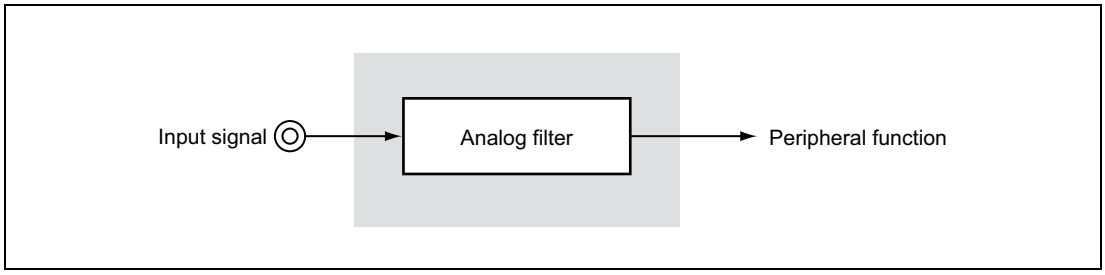



**Figure 2.13 Block Diagram of Analog Filter Type C**

The generated signals are always input signals that have passed through an analog filter.

#### Analog filter type C in Standby mode

Pins equipped with type C analog filters in this product do not support the input of event signals to trigger wake-up from standby.

## 2.13.3 Digital Filters

### 2.13.3.1 Digital Filter Characteristic

The digital filters allow the filter characteristics to be adjusted accordingly to the needs of the application.

The input signal is sampled with the sampling frequency fs.

If a specified number of successive samples yield the same (high or low) level, the signal level is judged as valid and the filter output signal is set accordingly.

If an external signal level change is detected within the specified number of samples (same level samples), the signal level is judged as noise and the filter output signal does not change.

The length of an external signal pulse to be judged as noise depends on the sampling frequency and the specified number of same level samples.

Both parameters can be specified:

- DNFA<name>CTL.DNFA<name>PRS[2:0] select the sampling frequency based on
  $fs = f_{DNFATCKI} / 2^{DNFA<name>PRS[2:0]}$
  where $f_{DNFATCKI}$ is the frequency of the DNFATCKI clock.
- DNFA<name>CTL.DNFA<name>NFSTS[1:0] determines the number of same level samples, "s", (2 to 5):

  s = DNFA<name>NFSTS[1:0] + 2

External signal pulses shorter than the following are suppressed at all times.

s ×1/fs

External signal pulses longer than the following are always judged as valid and are passed on to the filter output.

(s + 1) × 1/fs

External signal pulses in the following range may be suppressed or judged as valid.

s × 1/fs to (s + 1) × 1/fs

The filter operation is illustrated in the figure below with DNFA<name>NFSTS[1:0] = $01_B$, i.e. s = 3 same level samples.

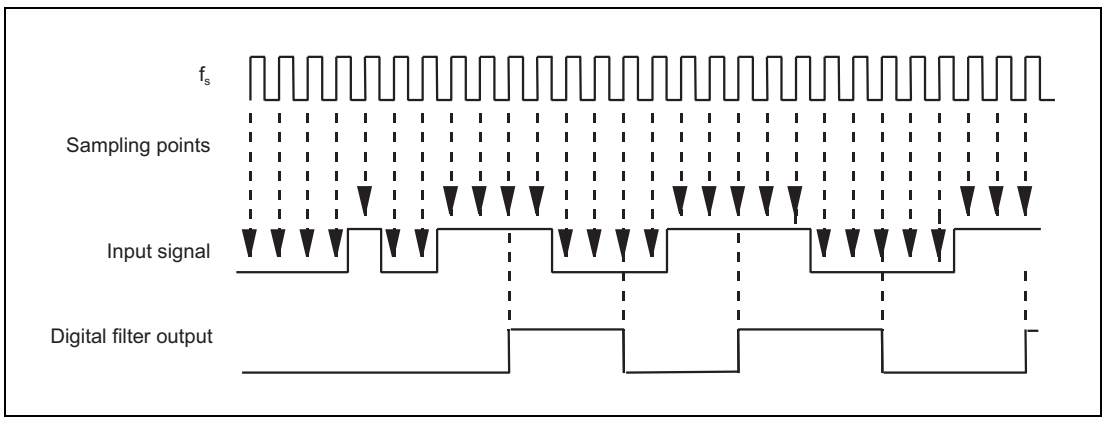



**Figure 2.14 Digital Filter Function**

### 2.13.3.2 Digital Filter Groups

The input signals processed through digital filters are ordered in groups of up to 16 signals.

The digital filter characteristics, specified by DNFA<name>CTL.DNFA<name>PRS[2:0] and DNFA<name>NFSTS[1:0] apply to the signals.

However, the digital filter for each signal can be enabled or disabled separately by DNFA<name>EN.DNFA<name>ENLm (m = 0 to 7) and DNFA<name>EN.DNFA<name>ENHm (m = 0 to 7).

**CAUTIONS**

1. **When the output signal from the digital filter is input to an alternative function, allow at least the following interval to elapse after the digital filter is enabled (DNFA<name>EN.DNFA<name>ENLm (m = 0 to 7) = 1 and DNFA<name>EN.DNFA<name>ENHm (m = 0 to 7) = 1) for the port pin to switch to the alternative function.**
   **s = DNFA<name>NFSTS[1:0] + 2**
   **$s \times 1/f_S + 2 \times 1/f_{DNFATCKI}$**
2. **When a digital filter's output signal is used as an interrupt signal, only enable the digital filter (DNFA<name>EN.DNFA<name>ENLm (m = 0 to 7) = 1 and DNFA<name>EN.DNFA<name>ENHm (m = 0 to 7) = 1) while interrupts are disabled. Furthermore, only enable interrupts after enabling the digital filter, waiting for the time below to elapse, and then clearing the interrupt request flag.**
   **$s \times 1/f_S + 3 \times 1/f_{DNFATCKI}$**

### 2.13.3.3 Digital Filters in Standby Mode

Digital filters for the function of waking-up from the DeepSTOP mode are located in the Always-On area (AWO area). Digital filters on the Always-On area (AWO area) are always operating.

Digital noise elimination requires the clock supply DNFATCKI to operate.
Pins equipped with digital filters in this product do not support the input of event signals to trigger wake-up from standby.

### 2.13.3.4 Digital Filter Control Registers

For each group consisting of up to 16 digital filters, the digital noise elimination control register DNFA<name>CTL and digital noise elimination enable register DNFA<name>EN are used to set all the filters in the same group (<name> = peripheral function group).

The DNFA<name>CTL register specifies the characteristics of the digital noise elimination filter for the digital filter of <name>.

The DNFA<name>EN register enables/disables each filter by setting the corresponding bit in DNFA<name>EN.DNFA<name>ENLm (m = 0 to 7) and DNFA<name>EN.DNFA<name>ENHm (m = 0 to 7).

The edge detection setup is done via the filter dedicated control register and the registers for individual peripheral functions.

The FCLA0CTL0_ADCn registers are ordered in groups of 3 registers with the same index n. The register index n is in 0 or 1.

The assignment of the input signals to the control registers and their addresses are given in **Table 2.75, Input Pins that Incorporate Digital Filter Type D** and **Table 2.76, Input Pins that Incorporate Digital Filter Type E** in **Section 2.12.1, Port Filter Assignment**.

**CAUTION**

**Do not change any control register settings while the corresponding digital filter is enabled by DNFA<name>EN.DNFA<name>ENLm (m = 0 to 7) = 1 and DNFA<name>EN.DNFA<name>ENHm (m = 0 to 7) = 1. Otherwise an unintended filter output may be generated.**

#### (1) Digital filter type D

A block diagram of digital filter type D is shown below.

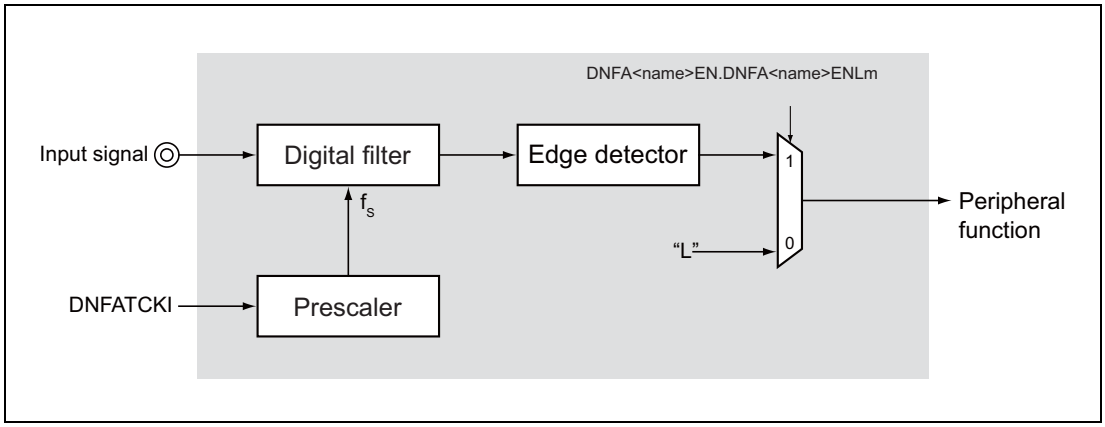



**Figure 2.15 Block Diagram of Digital Filter Type D**

The generated signal depends on the register setting, as shown in the following table.

**Table 2.79 Output Options for Digital Filter Type D**

| DNFA<name>EN.DNFA<name>ENLm | Signals Output to Peripheral Functions |
|---|---|
| 0 | Fixed to low level |
| 1 | Input signal passed through filter |

### (2) Digital filter type E

A block diagram of digital filter type E is shown below.

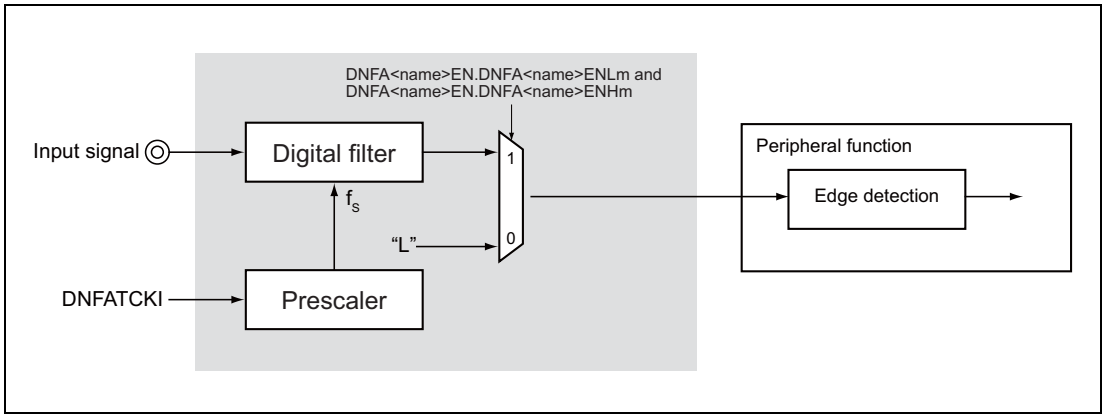



**Figure 2.16 Block Diagram of Digital Filter Type E**

The generated signal depends on the register setting, as shown in the following table.

**Table 2.80 Output Options for Digital Filter Type E**

| DNFA<name>EN.DNFA<name>ENLm and DNFA<name>EN.DNFA<name>ENHm | Signals Output to Peripheral Functions |
|---|---|
| 0 | Fixed to low level |
| 1 | Input signal passed through filter |

## 2.13.4 Filter Control Registers

The analog and digital filters are controlled and operated by the following registers:

**Table 2.81 List of Filter Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| FCLA0 | Filter control register m | FCLA0CTLm_<name> | The addresses are shown in the tables in **Section 2.12.1, Port Filter Assignment**. |
| DNF | Digital noise elimination control register | DNFA<name>CTL | |
| | Digital noise elimination enable register | DNFA<name>EN | |
| | Digital noise elimination enable H register | DNFA<name>ENH | |
| | Digital noise elimination enable L register | DNFA<name>ENL | |

### 2.13.4.1 FCLA0CTLm_<name> — Filter Control Register

This register controls the analog and digital filter operation.

**Access:** This register can be read or written in 8-bit units.

**Address:** The allocation of input signals to FCLA0CTLm_<name> registers and the address of each register are shown in the tables in **Section 2.12.1, Port Filter Assignment**.

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | FCLA0INTLm_<name> | FCLA0INTFm_<name> | FCLA0INTRm_<name> |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W |

**Table 2.82 FCLA0CTLm_<name> Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 | FCLA0INTLm_<name> | Detection Mode Selection<br>0: Edge detection<br>1: Level detection<br>**Note:** This bit is only valid for analog filter type A. |
| 1 | FCLA0INTFm_<name> | • In level detection mode (FCLA0INTLm_<name> = 1): This bit has no effect.<br>• In edge detection mode (FCLA0INTLm_<name> = 0): Falling edge detection control<br>0: Falling edge detection disabled<br>1: Falling edge detection enabled<br>**Note:** This bit is only valid for analog filter type A and digital filter type D. However, digital filter type D is placed in edge detection mode. |
| 0 | FCLA0INTRm_<name> | • In level detection mode (FCLA0INTLm_<name> = 1): Detected level selection<br>0: Low level detection<br>1: High level detection<br>• In edge detection mode (FCLA0INTLm_<name> = 0): Rising edge detection control<br>0: Rising edge detection disabled<br>1: Rising edge detection enabled<br>**Note:** This bit is only valid for analog filter type A and digital filter type D. However, digital filter type D is placed in edge detection mode. |

**CAUTION**

Digital filter type D: Always set bit 2 to "0".

### 2.13.4.2 DNFA<name>CTL — Digital Noise Elimination Control Register

This register is used to specify the filter characteristics of the digital noise elimination filter.

**NOTE**

This register is only valid for digital filter type D and digital filter type E.

**Access:** This register can be read or written in 8-bit units.

**Address:** For the correspondence between the DNFA<name>CTL register and input signals, and the addresses of individual registers, see **Table 2.75, Input Pins that Incorporate Digital Filter Type D** and **Table 2.76, Input Pins that Incorporate Digital Filter Type E** in **Section 2.12.1, Port Filter Assignment**.

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | DNFA<name>NFSTS[1:0] | | — | — | DNFA<name>PRS[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R | R | R/W | R/W | R/W |

**Table 2.83 DNFA<name>CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 6, 5 | DNFA<name>NFSTS[1:0] | The DNFA<name>NFSTS[1:0] bits specify the number of samples used to judge whether an external signal pulse is valid. |

| DNFA<name>NFSTS[1:0] | Number of Samples |
|---|---|
| $00_B$ | 2 |
| $01_B$ | 3 |
| $10_B$ | 4 |
| $11_B$ | 5 |

| Bit Position | Bit Name | Function |
|---|---|---|
| 4, 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 to 0 | DNFA<name>PRS[2:0] | Digital filter sampling clock selection |

| DNFA<name>PRS[2:0] | Sampling Clock Frequency |
|---|---|
| $000_B$ | DNFATCKI/1 |
| $001_B$ | DNFATCKI/2 |
| $010_B$ | DNFATCKI/4 |
| $011_B$ | DNFATCKI/8 |
| $100_B$ | DNFATCKI/16 |
| $101_B$ | DNFATCKI/32 |
| $110_B$ | DNFATCKI/64 |
| $111_B$ | DNFATCKI/128 |

### 2.13.4.3 DNFA<name>EN — Digital Noise Elimination Enable Register

This register enables and disables digital noise elimination for a specified input signal.

**NOTE**

This register is only valid for digital filter type D and digital filter type E.

**Access:** This register can be read or written in 16-bit units.
The upper- and lower-order bytes (DNFA<name>ENH[7:0] and DNFA<name>ENL[7:0]) are accessible in 8- or 1-bit units respectively by setting DNFA<name>ENH. and DNFA<name>ENL.

**Address:** For the correspondence between the DNFA<name>EN register and input signals, and the addresses of individual registers, see **Table 2.75, Input Pins that Incorporate Digital Filter Type D** and **Table 2.76, Input Pins that Incorporate Digital Filter Type E** in **Section 2.12.1, Port Filter Assignment**.

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DNFA<name>ENH7 | DNFA<name>ENH6 | DNFA<name>ENH5 | DNFA<name>ENH4 | DNFA<name>ENH3 | DNFA<name>ENH2 | DNFA<name>ENH1 | DNFA<name>ENH0 | DNFA<name>ENL7 | DNFA<name>ENL6 | DNFA<name>ENL5 | DNFA<name>ENL4 | DNFA<name>ENL3 | DNFA<name>ENL2 | DNFA<name>ENL1 | DNFA<name>ENL0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.84 DNFA<name>EN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | DNFA<name>ENH[7:0]<br>DNFA<name>ENL[7:0] | Digital Noise Elimination Enable/Disable Control<br>0: Digital noise elimination disabled<br>1: Digital noise elimination enabled |

### 2.13.4.4 DNFA<name>ENH – Digital Noise Elimination Enable H Register

Setting in this register correspond to those of the 8 upper-order bits of the DNFA<name>ENH register.

**NOTE**

This register is only valid for digital filter type E.

**Access:** This register can be read or written in 8-bit or 1-bit units.

**Address:** For the correspondence between the DNFA<name>ENH register and input signals, and the addresses of individual registers, see **Table 2.76, Input Pins that Incorporate Digital Filter Type E** in **Section 2.12.1, Port Filter Assignment**.

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DNFA <name> ENH7 | DNFA <name> ENH6 | DNFA <name> ENH5 | DNFA <name> ENH4 | DNFA <name> ENH3 | DNFA <name> ENH2 | DNFA <name> ENH1 | DNFA <name> ENH0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

For details of the respective bit functions, see **Section 2.13.4.3, DNFA<name>EN — Digital Noise Elimination Enable Register**.

### 2.13.4.5 DNFA<name>ENL – Digital Noise Elimination Enable L Register

Setting in this register correspond to those of the 8 lower-order bits of the DNFA<name>ENL register.

**NOTE**

This register is only valid for digital filter type D and digital filter type E.

**Access:** This register can be read or written in 8-bit or 1-bit units.

**Address:** For the correspondence between the DNFA<name>ENL register and input signals, and the addresses of individual registers, see **Table 2.75, Input Pins that Incorporate Digital Filter Type D** and **Table 2.76, Input Pins that Incorporate Digital Filter Type E** in **Section 2.12.1, Port Filter Assignment**.

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DNFA <name> ENL7 | DNFA <name> ENL6 | DNFA <name> ENL5 | DNFA <name> ENL4 | DNFA <name> ENL3 | DNFA <name> ENL2 | DNFA <name> ENL1 | DNFA <name> ENL0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

For details of the respective bit functions, see **Section 2.13.4.3, DNFA<name>EN — Digital Noise Elimination Enable Register**.

# Section 3 CPU System

## 3.1 Overview

### 3.1.1 Block Configuration

**Figure 3.1** shows the block configuration diagram of RH850/F1K.

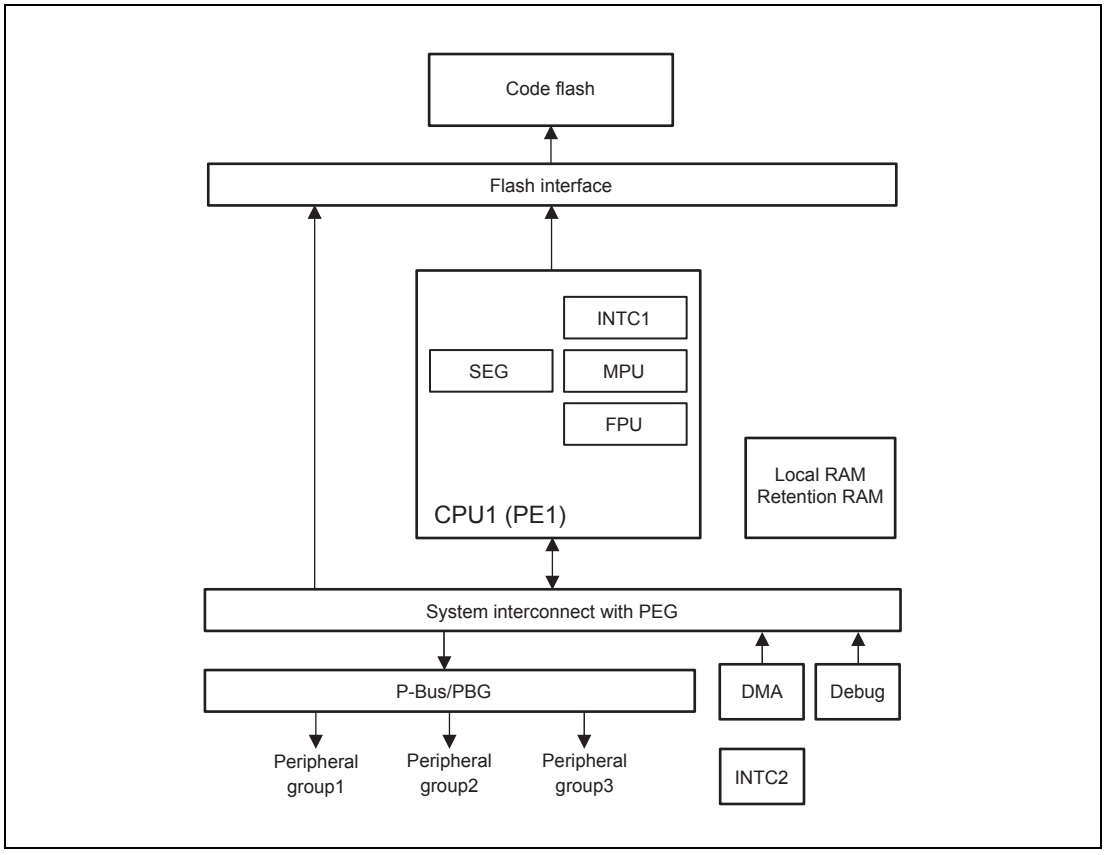



**Figure 3.1 Block Configuration Diagram of the RH850/F1K**

#### CPU1 (PE1)

The RH850 G3KH Core is used as the main CPU.

#### Local RAM

This is a high-speed accessible RAM

#### Retention RAM

The retention RAM is used to retain values in DeepSTOP mode. Since the continuous local RAM area is assigned for the retention RAM, the retention RAM can also serve as a local RAM for sharing data with the DMA.

#### Code flash

The code flash memory is included for program storage. It is connected with CPU1 via the flash interface.

### Data flash

The data flash memory can be rewritten by the CPU1. It has a greater write endurance than the code flash memory.

### P-Bus

The P-Bus connects the peripheral IPs. The P-Bus is divided into three peripheral groups, 1 to 3.

### INTC1, INTC2

There are two interrupt controllers, INTC1 and INTC2.

### DMA

The DMA transfer module (DMAC) is included.

### Slave guard

The slave guard is a function to prevent unauthorized access from the specific bus master, and consists of the following guard structures:

(1) PE guard (PEG)

The PE guard is a function to prevent unauthorized access to the resources (local RAM and retention RAM) in the PE from an external master. After reset is released, access from other than the own PE is prohibited.

(2) System error generation (SEG)

The registers control how to response SYSERR.

(3) Peripheral guard (PBG)

The peripheral guard is a function to prevent unauthorized access to peripherals. The control registers in the peripheral circuits are protected against illegal accesses.

For details, see **Section 33, Functional Safety**.

# 3.2 CPU

## 3.2.1 Core Functions

### 3.2.1.1 Features

**Table 3.1** lists features of the RH850 G3KH core.

**Table 3.1 Features of the RH850 G3KH Core**

| Item | Feature |
|---|---|
| CPU | • Advanced 32-bit architecture for embedded control<br>• 32-bit internal data bus<br>• Thirty-two 32-bit general-purpose registers<br>• RISC-type instruction set<br>– Long-/short-format load/store instructions<br>– Three-operand instructions<br>– Instruction set based on C language<br>• CPU operating modes<br>– User mode and supervisor mode<br>• Address space: 4-Gbyte linear address space for both data and instructions |
| Coprocessor | • Floating-point operation coprocessor (FPU)<br>– Supports single precision (32 bits)<br>– Supports data types and exceptions conforming to IEEE754.<br>– Rounding mode: Neighborhood, 0 direction, +∞ direction, and –∞ direction<br>– Handling of denormalized numbers: Rounding down to 0 or exception notification to conform to IEEE754 |
| Exception/ Interrupt | • 16 interrupt priority levels settable for each channel<br>• Vector selection method selectable according to performance request or memory usage<br>– Direct branching exception vectors<br>– Indirect branching exception vectors referring to the address table<br>• Supports the high-speed save/return processing of the context by the dedicated instructions (PUSHSP and POPSP) at the generation of an interrupt |
| Memory management | • Memory protection function (MPU): 16 areas settable |
| Cache | • No cache memory is equipped. |

### 3.2.1.2 Register Set

This subsection explains the program registers and system registers incorporated in this CPU.

#### (1) Program Registers

Program registers include the general-purpose registers (r0 to r31) and program counter (PC).

**Table 3.2 Program Registers**

| Program Register | Name | Function | Description |
|---|---|---|---|
| General-purpose registers | r0 | Zero register | Always retains “0” |
| | r1 | Assembler reserved register | Used as working register for generating addresses |
| | r2 | Register for address and data variables (used when the real-time OS used does not use this register) | |
| | r3 | Stack pointer (SP) | Used for generating a stack frame when a function is called |
| | r4 | Global pointer (GP) | Used for accessing a global variable in the data area |
| | r5 | Text pointer (TP) | Used as a register that indicates the start of the text area (area where program code is placed) |
| | r6 to r29 | Register for address and data variables | |
| | r30 | Element pointer (EP) | Used as a base pointer for generating addresses when accessing memory |
| | r31 | Link pointer (LP) | Used when the compiler calls a function |
| Program counter | PC | Retains instruction addresses during execution of programs | |

**NOTE**

For further descriptions of r1, r3 to r5, and r31 used by the assembler and/or C compiler, see the specification of each software development environment.

### (a) General-purpose Registers

A total of 32 general-purpose registers (r0 to r31) are provided. All of these registers can be used for either data variables or address variables. Of the general-purpose registers, r0 to r5, r30, and r31 are assumed to be used for special purposes in software development environments, so it is necessary to note the following when using them.

1. r0, r3, r30

   These registers are implicitly used by instructions.

   r0 is a register that always retains "0". It is used for operations that use 0 and addressing with base address being 0.

   r3 is implicitly used by the PREPARE, DISPOSE, PUSHSP, and POPSP instructions.

   r30 is used as a base pointer when the SLD or SST instruction accesses memory.

2. r1, r4, r5, r31

   These registers are implicitly used by the assembler and C compiler.

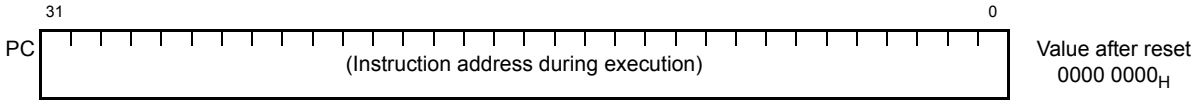
   When using these registers, register contents must first be saved so they are not lost and can be restored after the registers are used.

3. r2

   This register might be used by a real-time OS in some cases. If the real-time OS that is being used does not use r2, r2 can be used as a register for address variables or data variables.

### (b) PC – Program Counter

The PC retains the address of the instruction being executed. Bit 0 is fixed to 0, and branching to an odd number address is disabled.




**Note:** The value after reset differs depending on the setting value of the reset vector. For details, see **(q) RBASE — Reset Vector Base Address Register**.

### (2) Basic System Registers

The basic system registers are used to control CPU status and to retain exception information.

System registers are read from or written to by using the LDSR and STSR instructions and specifying the system register number, which is made up of a register number and a selection ID.

Table 3.3 Basic System Registers

| Register No. (regID, selID) | Symbol | Function | Access Permission |
|---|---|---|---|
| SR0, 0 | EIPC | Status save registers when acknowledging EI level exception | SV |
| SR1, 0 | EIPSW | Status save registers when acknowledging EI level exception | SV |
| SR2, 0 | FEPC | Status save registers when acknowledging FE level exception | SV |
| SR3, 0 | FEPSW | Status save registers when acknowledging FE level exception | SV |
| SR5, 0 | PSW | Program status word | *1 |
| SR6, 0 | FPSR | (Refer to FPU function registers.) | CU and SV |
| SR7, 0 | FPEPC | (Refer to FPU function registers.) | CU and SV |
| SR8, 0 | FPST | (Refer to FPU function registers.) | CU |
| SR9, 0 | FPCC | (Refer to FPU function registers.) | CU |
| SR10, 0 | FPCFG | (Refer to FPU function registers.) | CU |
| SR11, 0 | FPEC | (Refer to FPU function registers.) | CU and SV |
| SR13, 0 | EIIC | EI level exception cause | SV |
| SR14, 0 | FEIC | FE level exception cause | SV |
| SR16, 0 | CTPC | CALLT execution status save register | UM |
| SR17, 0 | CTPSW | CALLT execution status save register | UM |
| SR20, 0 | CTBP | CALLT base pointer | UM |
| SR28, 0 | EIWR | EI level exception working register | SV |
| SR29, 0 | FEWR | FE level exception working register | SV |
| SR0, 1 | MCFG0 | Machine configuration | SV |
| SR2, 1 | RBASE | Reset vector base address | SV |
| SR3, 1 | EBASE | Exception handler vector address | SV |
| SR4, 1 | INTBP | Base address of the interrupt handler address table | SV |
| SR5, 1 | MCTL | CPU control | SV |
| SR6, 1 | PID | Processor ID | SV |
| SR11, 1 | SCCFG | SYSCALL operation setting | SV |
| SR12, 1 | SCBP | SYSCALL base pointer | SV |
| SR0, 2 | HTCFG0 | Thread configuration | SV |
| SR6, 2 | MEA | Memory error address | SV |
| SR7, 2 | ASID | Address space ID | SV |
| SR8, 2 | MEI | Memory error information | SV |

Note 1. The access permission differs depending on the bit.

(a) EIPC — Status Save Register when Acknowledging EI Level Exception

When an EI level exception is acknowledged, the address of the instruction that was being executed when the EI level exception occurred, or of the next instruction, is saved to the EIPC register (see "*Types of Exceptions*" in *Software Manual*).

Because there is only one pair of EI level exception status save registers, when processing multiple exceptions, the contents of these registers must be saved by a program.

Be sure to set an even-numbered address to the EIPC register. An odd-numbered address cannot be specified.

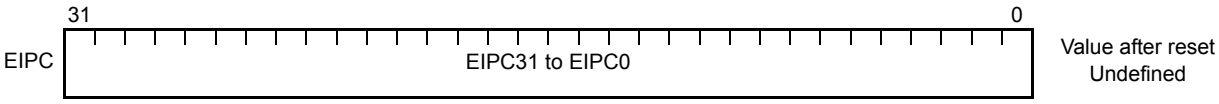



**Table 3.4 EIPC Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 1 | EIPC31 to EIPC1 | These bits indicate the PC saved when an EI level exception is acknowledged. | R/W | Undefined |
| 0 | EIPC0 | This bit indicates the PC saved when an EI level exception is acknowledged. Always set this bit to 0. Even if it is set to 1, the value transferred to the PC when the EIRET instruction is executed is 0. | R/W | Undefined |

(b) EIPSW — Status Save Register when Acknowledging EI Level Exception

When an EI level exception is acknowledged, the current PSW setting is saved to the EIPSW register.

Because there is only one pair of EI level exception status save registers, when processing multiple exceptions, the contents of these registers must be saved by a program.

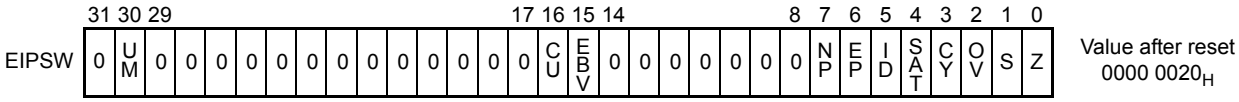



**Table 3.5 EIPSW Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 30 | UM | This bit stores the PSW.UM bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 29 to 17 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 16 | CU | This bit stores the PSW.CU field setting when an EI level exception is acknowledged. | R/W | 0 |
| 15 | EBV | This bit stores the PSW.EBV bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 14 to 8 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 7 | NP | This bit stores the PSW.NP bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 6 | EP | This bit stores the PSW.EP bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 5 | ID | This bit stores the PSW.ID bit setting when an EI level exception is acknowledged. | R/W | 1 |
| 4 | SAT | This bit stores the PSW.SAT bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 3 | CY | This bit stores the PSW.CY bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 2 | OV | This bit stores the PSW.OV bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 1 | S | This bit stores the PSW.S bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 0 | Z | This bit stores the PSW.Z bit setting when an EI level exception is acknowledged. | R/W | 0 |

(c) FEPC — Status Save Register when Acknowledging FE Level Exception

When an FE level exception is acknowledged, the address of the instruction that was being executed when the FE level exception occurred, or of the next instruction, is saved to the FEPC register (see *"Types of Exceptions"* in *Software Manual*).

Because there is only one pair of FE level exception status save registers, when processing multiple exceptions, the contents of these registers must be saved by a program.

Be sure to set an even-numbered address to the FEPC register. An odd-numbered address cannot be specified.

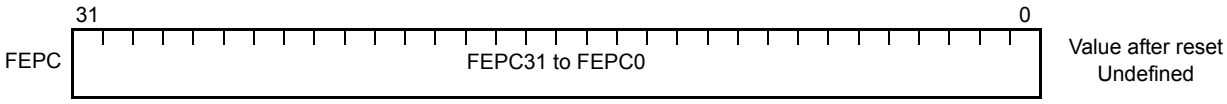



**Table 3.6 FEPC Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 1 | FEPC31 to FEPC1 | These bits indicate the PC saved when an FE level exception is acknowledged. | R/W | Undefined |
| 0 | FEPC0 | This bit indicates the PC saved when an FE level exception is acknowledged.<br>Always set this bit to 0. Even if it is set to 1, the value transferred to the PC when the FERET instruction is executed is 0. | R/W | Undefined |

(d) FEPSW — Status Save Register when Acknowledging FE Level Exception

When an FE level exception is acknowledged, the current PSW setting is saved to the FEPSW register.

Because there is only one pair of FE level exception status save registers, when processing multiple exceptions, the contents of these registers must be saved by a program.

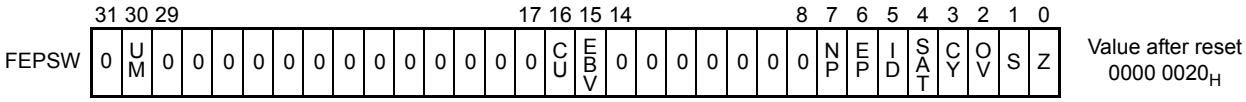



**Table 3.7 FEPSW Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 30 | UM | This bit stores the PSW.UM bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 29 to 17 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 16 | CU | This bit stores the PSW.CU field setting when an FE level exception is acknowledged. | R/W | 0 |
| 15 | EBV | This bit stores the PSW.EBV bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 14 to 8 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 7 | NP | This bit stores the PSW.NP bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 6 | EP | This bit stores the PSW.EP bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 5 | ID | This bit stores the PSW.ID bit setting when an FE level exception is acknowledged. | R/W | 1 |
| 4 | SAT | This bit stores the PSW.SAT bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 3 | CY | This bit stores the PSW.CY bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 2 | OV | This bit stores the PSW.OV bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 1 | S | This bit stores the PSW.S bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 0 | Z | This bit stores the PSW.Z bit setting when an FE level exception is acknowledged. | R/W | 0 |

(e) PSW — Program Status Word

PSW (program status word) is a set of flags that indicate the program status (instruction execution result) and bits that indicate the operation status of the CPU (flags are bits in the PSW that are referenced by conditional instructions (Bcond, CMOV, etc.)).

**CAUTIONS**

1. **When the LDSR instruction is used to change the contents of each bit in this register, the changed contents become valid immediately after completion of the LDSR instruction execution.**
2. **The access permission for the PSW register differs depending on the bit. All bits can be read, but some bits can only be written under certain conditions. See Table 3.8, Access Permission for PSW Register for the access permission for each bit.**

**Table 3.8 Access Permission for PSW Register**

| Bit | | Access Permission When Reading | Access Permission When Writing |
|---|---|---|---|
| 30 | UM | UM | SV*1 |
| 16 | CU | UM | SV*1 |
| 15 | EBV | UM | SV*1 |
| 7 | NP | UM | SV*1 |
| 6 | EP | UM | SV*1 |
| 5 | ID | UM | SV*1 |
| 4 | SAT | UM | UM |
| 3 | CY | UM | UM |
| 2 | OV | UM | UM |
| 1 | S | UM | UM |
| 0 | Z | UM | UM |

Note 1. The access permission for the whole PSW register is UM, so the PIE exception does not occur even if the register is written by using an LDSR instruction when PSW.UM is 1. In this case, writing is ignored.

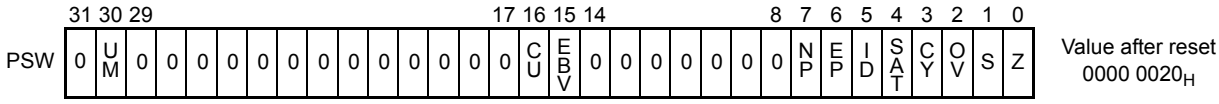



**Table 3.9 PSW Register Contents (1/2)**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 30 | UM | This bit indicates that the CPU is in user mode (in UM mode).<br>0: Supervisor mode<br>1: User mode | R/W | 0 |
| 29 to 17 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 16 | CU | This bit indicates the coprocessor use permissions. When the bit corresponding to the coprocessor is 0, a coprocessor unusable exception occurs if an instruction for the coprocessor is executed or a coprocessor resource (system register) is accessed.<br>CU bit 16: FPU | R/W | 0 |

Table 3.9 PSW Register Contents (2/2)

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 15 | EBV | This bit indicates the reset vector and exception vector operation. See the description on RBASE ((q) RBASE — Reset Vector Base Address Register) and EBASE ((r) EBASE — Exception Handler Vector Address Register) in this section. | R/W | 0 |
| 14 to 8 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 7 | NP | This bit disables the acknowledgement of FE level exception. When an FE level exception is acknowledged, this bit is set to 1 to disable the acknowledgement of EI level and FE level exceptions. As for the exceptions which the NP bit disables the acknowledgment, see **Table 7.1, List of Exception Sources**.<br>0: The acknowledgement of FE level exception is enabled.<br>1: The acknowledgement of FE level exception is disabled. | R/W | 0 |
| 6 | EP | This bit indicates that an exception other than an interrupt controlled by the interrupt controller is being serviced. It is set to 1 when the corresponding exception occurs. This bit does not affect acknowledging an exception request even when it is set to 1.<br>0: An exception other than an interrupt is not being serviced.<br>1: An exception other than an interrupt is being serviced. | R/W | 0 |
| 5 | ID | This bit disables the acknowledgement of EI level exception. When an EI level or FE level exception is acknowledged, this bit is set to 1 to disable the acknowledgement of EI level exception. As for the exceptions which the ID bit disables the acknowledgment, see **Table 7.1, List of Exception Sources**. This bit is also used to disable EI level exceptions from being acknowledged as a critical section while an ordinary program or interrupt is being serviced. It is set to 1 when the DI instruction is executed, and cleared to 0 when the EI instruction is executed. The change of the ID bit by the EI or ID instruction will be enabled from the next instruction.<br>0: EI level exception is not being processed or the section is not a critical section (after execution of EI instruction).<br>1: EI level exception is being processed or the section is a critical section (after execution of DI instruction). | R/W | 1 |
| 4 | SAT*1 | This bit indicates that the operation result is saturated because the operation result of a saturated operation instruction has overflowed. This is a cumulative flag, so when the operation result of the saturated operation instruction becomes saturated, this bit is set to 1, but it is not cleared to 0 when the operation result for a subsequent instruction is not saturated. This bit is cleared to 0 by the LDSR instruction. This bit is neither set to 1 nor cleared to 0 when an arithmetic operation instruction is executed.<br>0: Not saturated<br>1: Saturated | R/W | 0 |
| 3 | CY | This bit indicates whether a carry or borrow has occurred in the operation result.<br>0: Carry and borrow have not occurred.<br>1: Carry or borrow has occurred. | R/W | 0 |
| 2 | OV*1 | This bit indicates whether or not an overflow has occurred during an operation.<br>0: Overflow has not occurred.<br>1: Overflow has occurred. | R/W | 0 |
| 1 | S*1 | This bit indicates whether or not the result of an operation is negative.<br>0: Result of operation is positive or 0.<br>1: Result of operation is negative. | R/W | 0 |
| 0 | Z | This bit indicates whether or not the result of an operation is 0.<br>0: Result of operation is not 0.<br>1: Result of operation is 0. | R/W | 0 |

Note 1. The operation result of the saturation processing is determined in accordance with the contents of the OV flag and S flag during a saturated operation. The SAT flag is set to 1 only when the OV flag is set to 1 in a saturated operation.

| Operation Result Status | Flag Status | | | Operation Result after Saturation Processing |
|---|---|---|---|---|
| | SAT | OV | S | |
| Exceeded positive maximum value | 1 | 1 | 0 | 7FFF FFFF$_H$ |
| Exceeded negative maximum value | 1 | 1 | 1 | 8000 0000$_H$ |
| Positive (maximum value not exceeded) | Value prior to operation is retained. | 0 | 0 | Operation result itself |
| Negative (maximum value not exceeded) | | | 1 | |

### (f) EIIC — EI Level Exception Source Register

The EIIC register retains the source of any EI level exception that occurs. The value retained in this register is an exception source code corresponding to a specific exception source.

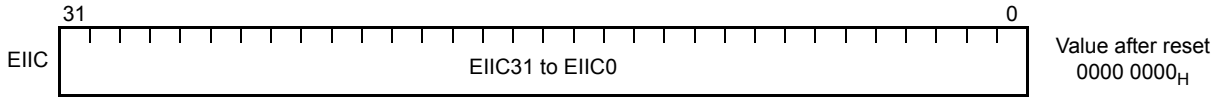



**Table 3.10 EIIC Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 0 | EIIC31 to EIIC0 | These bits store the exception source code when an EI level exception is acknowledged.<br>The EIIC15 to EIIC0 field stores the lower 16 bits of the exception source code.<br>The EIIC31 to EIIC16 field stores detailed exception source codes defined individually for each exception. If there is no particular definition of a function related to the exception, these bits are set to 0. | R/W | 0 |

### (g) FEIC — FE Level Exception Source Register

The FEIC register retains the source of any FE level exception that occurs. The value retained in this register is an exception source code corresponding to a specific exception source.

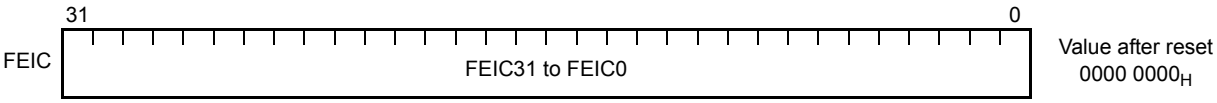



**Table 3.11 FEIC Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 0 | FEIC31 to FEIC0 | These bits store the exception source code when an FE level exception is acknowledged.<br>The FEIC15 to FEIC0 field stores the lower 16 bits of the exception source code.<br>The FEIC31 to FEIC16 field stores detailed exception source codes defined individually for each exception. If there is no particular definition of a function related to the exception, these bits are set to 0. | R/W | 0 |

### (h) CTPC — Status Save Register when Executing CALLT Instruction

When a CALLT instruction is executed, the address of the next instruction after the CALLT instruction is saved to CTPC. Be sure to set an even-numbered address to the CTPC register. An odd-numbered address cannot be specified.

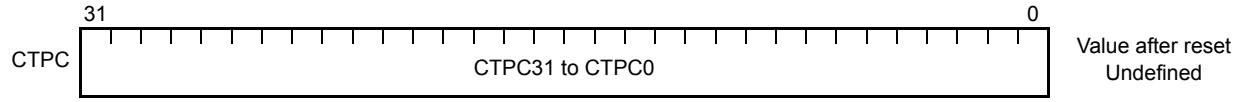



**Table 3.12 CTPC Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 1 | CTPC31 to CTPC1 | These bits indicate the PC of the instruction after the CALLT instruction. | R/W | Undefined |
| 0 | CTPC0 | This bit indicates the PC of the instruction after the CALLT instruction. Always set this bit to 0. Even if it is set to 1, the value transferred to the PC when the CTRET instruction is executed is 0. | R/W | Undefined |

### (i) CTPSW — Status Save Register when Executing CALLT Instruction

When a CALLT instruction is executed, some of the PSW (program status word) settings are saved to CTPSW.

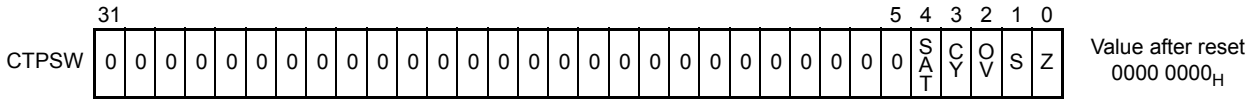



**Table 3.13 CTPSW Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 5 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 4 | SAT | This bit stores the PSW.SAT bit setting when the CALLT instruction is executed. | R/W | 0 |
| 3 | CY | This bit stores the PSW.CY bit setting when the CALLT instruction is executed. | R/W | 0 |
| 2 | OV | This bit stores the PSW.OV bit setting when the CALLT instruction is executed. | R/W | 0 |
| 1 | S | This bit stores the PSW.S bit setting when the CALLT instruction is executed. | R/W | 0 |
| 0 | Z | This bit stores the PSW.Z bit setting when the CALLT instruction is executed. | R/W | 0 |

### (j) CTBP — CALLT Base Pointer Register

The CTBP register is used to specify table addresses of the CALLT instruction and generate target addresses.

Be sure to set the CTBP register to a halfword address.

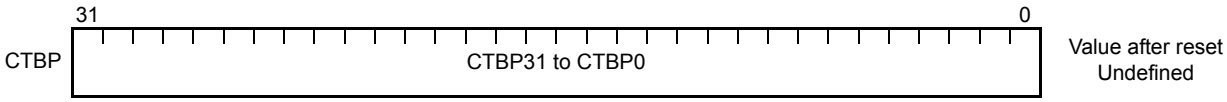



**Table 3.14 CTBP Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 1 | CTBP31 to CTBP1 | These bits indicate the base pointer address of the CALLT instruction. These bits indicate the start address of the table used by the CALLT instruction. | R/W | Undefined |
| 0 | CTBP0 | This bit indicates the base pointer address of the CALLT instruction. This bit indicates the start address of the table used by the CALLT instruction. Always set this bit to 0. | R | 0 |

### (k) ASID — Address Space ID Register

This register indicates the address space ID. This is used to identify the address space provided by the memory management function.

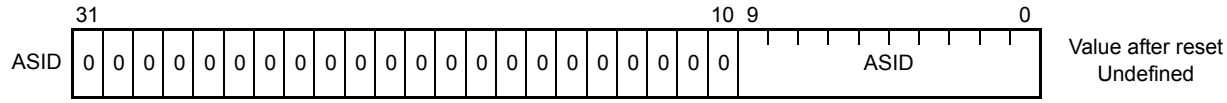



**Table 3.15 ASID Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 10 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 9 to 0 | ASID | These bits indicate the address space ID. | R/W | Undefined |

### (l) EIWR — EI Level Exception Working Register

The EIWR register is used as a working register when an EI level exception has occurred.

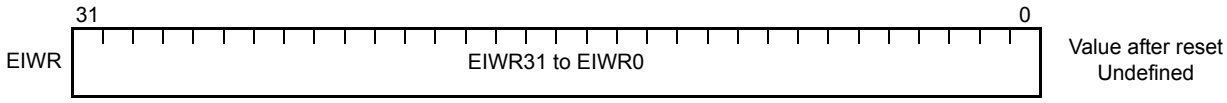



**Table 3.16 EIWR Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 0 | EIWR31 to EIWR0 | These bits constitute a working register that can be used for any purpose during the servicing of an EI level exception. This register can be used to temporarily save the values of general-purpose registers, etc. | R/W | Undefined |

(m) FEWR — FE Level Exception Working Register

The FEWR register is used as a working register when an FE level exception has occurred.

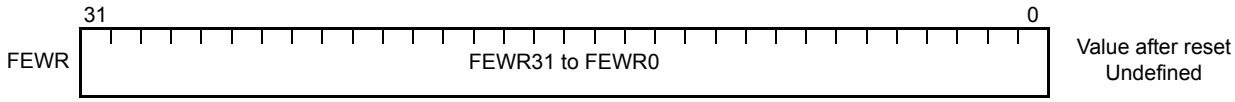



**Table 3.17 FEWR Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 0 | FEWR31 to FEWR0 | These bits constitute a working register that can be used for any purpose during the servicing of an FE level exception. This register can be used to temporarily save the values of general-purpose registers, etc. | R/W | Undefined |

(n) HTCFG0 — Thread Configuration Register

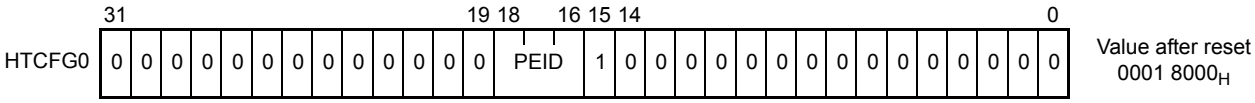



**Table 3.18 HTCFG0 Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 19 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 18 to 16 | PEID | These bits indicate the processor element number. | R | $001_B$ |
| 15 | — | (Reserved for future expansion. Be sure to set to 1.) | R | 1 |
| 14 to 0 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |

(o) MEA — Memory Error Address Register

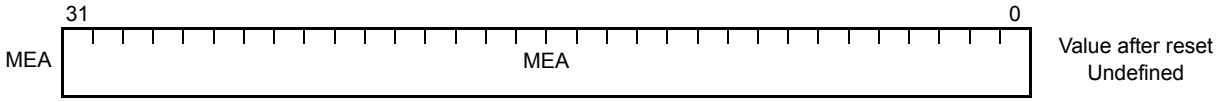



**Table 3.19 MEA Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 0 | MEA | These bits store the violation address when an MAE (misaligned) or MPU occurs. | R/W | Undefined |

(p) MEI — Memory Error Information Register

This register is used to store information about the instruction that caused a misaligned (MAE) or memory protection (MDP) exception when such an exception occurred. This information is used during emulation.

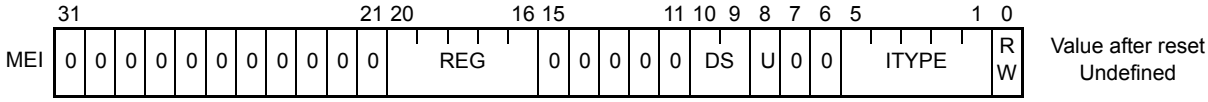



**Table 3.20 MEI Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 21 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 20 to 16 | REG | These bits indicate the number of the source or destination register accessed by the instruction that caused the exception. For details, see **Table 3.21**. | R/W | Undefined |
| 15 to 11 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 10, 9 | DS | These bits indicate the data type of the instruction that caused the exception.*1<br>0: Byte (8 bits)<br>1: Halfword (16 bits)<br>2: Word (32 bits)<br>3: Double-word (64 bits)<br>For details, see **Table 3.21**. | R/W | Undefined |
| 8 | U | This bit indicates the sign extension method of the instruction that caused the exception.<br>0: Signed<br>1: Unsigned<br>For details, see **Table 3.21**. | R/W | Undefined |
| 7, 6 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 5 to 1 | ITYPE | These bits indicate the instruction that caused the exception. For details, see **Table 3.21**. | R/W | Undefined |
| 0 | RW | This bit indicates whether the operation of the instruction that caused the exception was read (Load-memory) or write (Store-memory).<br>0: Read (Load-memory)<br>1: Write (Store-memory)<br>For details, see **Table 3.21**. | R/W | Undefined |

Note 1. Even if the data is divided and access is made several times due to the specifications of the hardware, the original data type indicated by the instruction is stored.

**Table 3.21 Instructions Causing Exceptions and Values of MEI Register (1/2)**

| Instruction | REG | DS | U | RW | ITYPE |
|---|---|---|---|---|---|
| SLD.B | dst | 0 (byte) | 0 (signed) | 0 (read) | $00000_B$ |
| SLD.BU | dst | 0 (byte) | 1 (unsigned) | 0 (read) | $00000_B$ |
| SLD.H | dst | 1 (halfword) | 0 (signed) | 0 (read) | $00000_B$ |
| SLD.HU | dst | 1 (halfword) | 1 (unsigned) | 0 (read) | $00000_B$ |
| SLD.W | dst | 2 (word) | 0 (signed) | 0 (read) | $00000_B$ |
| SST.B | src | 0 (byte) | 0 (signed) | 1 (write) | $00000_B$ |
| SST.H | src | 1 (halfword) | 0 (signed) | 1 (write) | $00000_B$ |
| SST.W | src | 2 (word) | 0 (signed) | 1 (write) | $00000_B$ |
| LD.B (disp16) | dst | 0 (byte) | 0 (signed) | 0 (read) | $00001_B$ |
| LD.BU (disp16) | dst | 0 (byte) | 1 (unsigned) | 0 (read) | $00001_B$ |
| LD.H (disp16) | dst | 1 (halfword) | 0 (signed) | 0 (read) | $00001_B$ |

Table 3.21 Instructions Causing Exceptions and Values of MEI Register (2/2)

| Instruction | REG | DS | U | RW | ITYPE |
|---|---|---|---|---|---|
| LD.HU (disp16) | dst | 1 (halfword) | 1 (unsigned) | 0 (read) | $00001_B$ |
| LD.W (disp16) | dst | 2 (word) | 0 (signed) | 0 (read) | $00001_B$ |
| ST.B (disp16) | src | 0 (byte) | 0 (signed) | 1 (write) | $00001_B$ |
| ST.H (disp16) | src | 1 (halfword) | 0 (signed) | 1 (write) | $00001_B$ |
| ST.W (disp16) | src | 2 (word) | 0 (signed) | 1 (write) | $00001_B$ |
| LD.B (disp23) | dst | 0 (byte) | 0 (signed) | 0 (read) | $00010_B$ |
| LD.BU (disp23) | dst | 0 (byte) | 1 (unsigned) | 0 (read) | $00010_B$ |
| LD.H (disp23) | dst | 1 (halfword) | 0 (signed) | 0 (read) | $00010_B$ |
| LD.HU (disp23) | dst | 1 (halfword) | 1 (unsigned) | 0 (read) | $00010_B$ |
| LD.W (disp23) | dst | 2 (word) | 0 (signed) | 0 (read) | $00010_B$ |
| ST.B (disp23) | src | 0 (byte) | 0 (signed) | 1 (write) | $00010_B$ |
| ST.H (disp23) | src | 1 (halfword) | 0 (signed) | 1 (write) | $00010_B$ |
| ST.W (disp23) | src | 2 (word) | 0 (signed) | 1 (write) | $00010_B$ |
| LD.DW (disp23) | dst | 3 (double-word) | 0 (signed) | 0 (read) | $00010_B$ |
| ST.DW (disp23) | src | 3 (double-word) | 0 (signed) | 1 (write) | $00010_B$ |
| LDL.W | dst | 2 (word) | 0 (signed) | 0 (read) | $00111_B$ |
| STC.W | src | 2 (word) | 0 (signed) | 1 (write) | $00111_B$ |
| CAXI | dst | 2 (word) | 0 (signed) | 0 (read)/1 (write) | $01000_B$ |
| SET1 | — | 0 (byte) | 0 (signed) | 0 (read)/1 (write) | $01001_B$ |
| CLR1 | — | 0 (byte) | 0 (signed) | 0 (read)/1 (write) | $01001_B$ |
| NOT1 | — | 0 (byte) | 0 (signed) | 0 (read)/1 (write) | $01001_B$ |
| TST1 | — | 0 (byte) | 0 (signed) | 0 (read) | $01001_B$ |
| PREPARE | — | 2 (word) | 0 (signed) | 1 (write) | $01100_B$ |
| DISPOSE | — | 2 (word) | 0 (signed) | 0 (read) | $01100_B$ |
| PUSHSP | — | 2 (word) | 0 (signed) | 1 (write) | $01101_B$ |
| POPSP | — | 2 (word) | 0 (signed) | 0 (read) | $01101_B$ |
| SWITCH | — | 1 (halfword) | 0 (signed) | 0 (read) | $10000_B$ |
| CALLT | — | 1 (halfword) | 1 (unsigned) | 0 (read) | $10001_B$ |
| SYSCALL | — | 2 (word) | 0 (signed) | 0 (read) | $10010_B$ |
| CACHE | — | — | — | — | — |
| Interrupt (table reference)*1 | — | 2 (word) | 0 (signed) | 0 (read) | $10101_B$ |

Note 1. When reading the interrupt vector by using the table reference method.

**NOTE**

dst: destination register number, src: source register number

### (q) RBASE — Reset Vector Base Address Register

This register indicates the reset vector address when there is a reset. If the PSW.EBV bit is 0, this vector address is also used as the exception vector address.

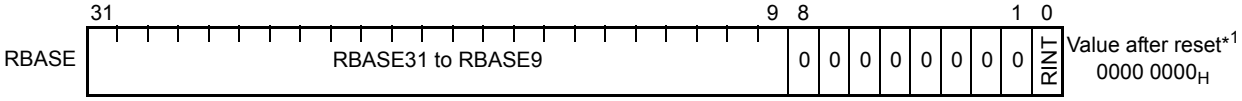



**Table 3.22 RBASE Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 9 | RBASE31 to RBASE9 | These bits indicate the reset vector when there is a reset. When PSW.EBV = 0, this address is also used as the exception vector.<br>For RBASE8 to RBASE0, 0 is used implicitly. | R | 0000 0000 0000 0000 0000 000B*1 |
| 8 to 1 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 0 | RINT | When the RINT bit is set, the exception handler address for interrupt servicing is reduced. See **Section 7.10.1, Direct Vector Method**. This bit is valid when PSW.EBV = 0. | R | 0 |

Note 1. The value depends on the reset vector. The values set at shipment are shown in the table. When the reset vector is modified, the address will be changed.

### (r) EBASE — Exception Handler Vector Address Register

This register indicates the exception handler vector address. This register is valid when the PSW.EBV bit is 1.

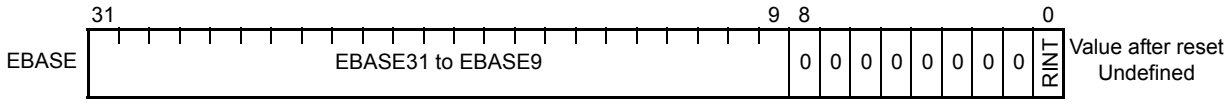



**Table 3.23 EBASE Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 9 | EBASE31 to EBASE9 | The exception handler routine address is changed to the address resulting from adding the offset address of each exception to the base address specified for this register.<br>For EBASE8 to EBASE0, 0 is used implicitly. | R/W | Undefined |
| 8 to 1 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 0 | RINT | When the RINT bit is set, the exception handler address for interrupt servicing is reduced. See **Section 7.10.1, Direct Vector Method**. | R/W | Undefined |

### (s) INTBP — Base Address Register of the Interrupt Handler Address Table

This register indicates the base address of the address table when the table reference method is selected as the interrupt handler address selection method.

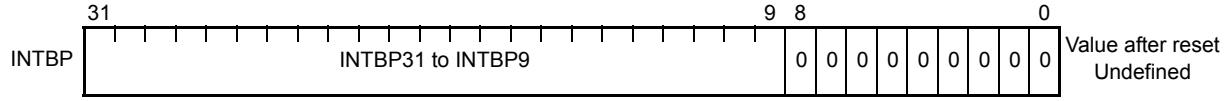



**Table 3.24 INTBP Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 9 | INTBP31 to INTBP9 | These bits indicate the base pointer address for an interrupt when the table reference method is used.<br>The value indicated by these bits is the first address in the table used to determine the exception handler when the interrupt specified by the table reference method (EIINT0 to EIINT511) is acknowledged.<br>For INTBP8 to INTBP0, 0 is used implicitly. | R/W | Undefined |
| 8 to 0 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |

### (t) PID — Processor ID Register

The PID register retains a processor identifier that is unique to the CPU. The PID register is a read-only register.

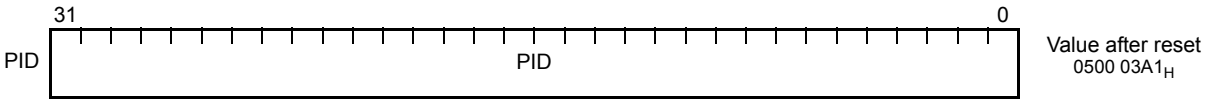



**Table 3.25 PID Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 24 | PID | Architecture Identifier<br>This identifier indicates the architecture of the processor. | R | $05_H$ |
| 23 to 8 | | Function Identifier<br>This identifier indicates the functions of the processor.<br>These bits indicate whether or not functions defined per bit are implemented (1: implemented, 0: not implemented).<br>Bits 23 to 11: Reserved<br>Bit 10: Double-precision floating-point operation function<br>Bit 9: Single-precision floating-point operation function<br>Bit 8: Memory protection function (MPU) | R | $0003_H$ |
| 7 to 0 | | Version Identifier<br>This identifier indicates the version of the processor. | R | $A1_H$ |

(u) SCCFG — SYSCALL Operation Setting Register

This register is used to specify operations related to the SYSCALL instruction. Be sure to set an appropriate value to this register before using the SYSCALL instruction.

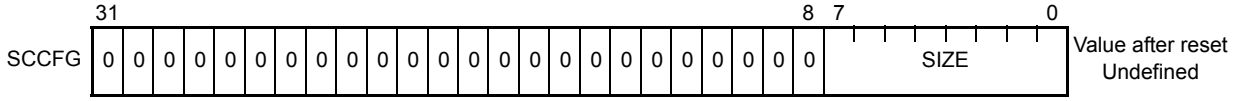



Table 3.26 SCCFG Register Contents

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 8 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 7 to 0 | SIZE | These bits specify the maximum number of entries of a table that the SYSCALL instruction references. The maximum number of entries the SYSCALL instruction references is 1 if SIZE is 0, and 256 if SIZE is 255.<br>By setting the maximum number of entries appropriately in accordance with the number of functions branched by the SYSCALL instruction, the memory area can be effectively used.<br>If vectors exceeding the maximum number of entries are specified for the SYSCALL instruction, the first entry is selected. Place an error processing routine at the first entry. | R/W | Undefined |

(v) SCBP — SYSCALL Base Pointer Register

The SCBP register is used to specify a table address of the SYSCALL instruction and generate a target address. Be sure to set an appropriate value to this register before using the SYSCALL instruction.

Be sure to set a word address to the SCBP register.

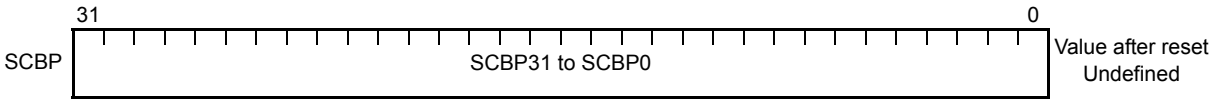



Table 3.27 SCBP Register Contents

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 2 | SCBP31 to SCBP2 | These bits indicate the base pointer address of the SYSCALL instruction.<br>These bits indicate the start address of the table used by the SYSCALL instruction. | R/W | Undefined |
| 1, 0 | SCBP1, SCBP0 | These bits indicate the base pointer address of the SYSCALL instruction.<br>These bits indicate the start address of the table used by the SYSCALL instruction.<br>Always set these bits to 0. | R | 0 |

(w) MCFG0 — Machine Configuration Register

This register indicates the CPU configuration.

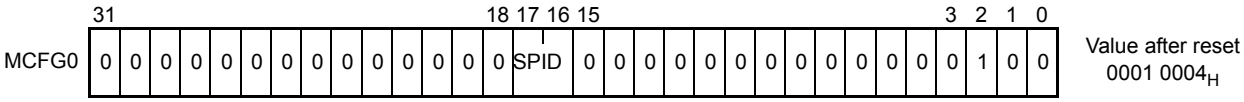



**Table 3.28 MCFG0 Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 18 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 17, 16 | SPID | These bits indicate the system protection number. | R/W | $01_B$ |
| 15 to 3 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 2 | — | (Reserved for future expansion. Be sure to set to 1.) | R | 1 |
| 1, 0 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |

(x) MCTL — Machine Control Register

This register is used to control the CPU.

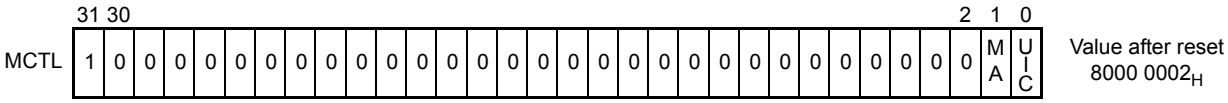



**Table 3.29 MCTL Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 | — | (Reserved for future expansion. Be sure to set to 1.) | R | 1 |
| 30 to 2 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 1 | MA | This bit is used to control the misaligned access.<br>0: In the event of a misaligned access, an exception is always generated.*1<br>1: The correct operation is controlled by hardware.*2 | R/W | 1 |
| 0 | UIC | This bit is used to control the interrupt enable/disable operation in user mode. When this bit is set to 1, executing the EI/DI instruction in user mode becomes possible. | R/W | 0 |

Note 1. Excluding LD.DW, and ST.DW for word boundary allocation.

Note 2. Exception still occurs in case of LD.DW or ST.DW for misaligned access except word boundary allocation.

### (3) Interrupt Function Registers

Table 3.30 Interrupt Function System Registers

| Register No. (regID, selID) | Symbol | Function | Access Permission |
|---|---|---|---|
| SR7, 1 | FPIPR | FPI exception interrupt priority setting | SV |
| SR10, 2 | ISPR | Priority of interrupt being serviced | SV |
| SR11, 2 | PMR | Interrupt priority masking | SV |
| SR12, 2 | ICSR | Interrupt control status | SV |
| SR13, 2 | INTCFG | Interrupt function setting | SV |

#### (a) FPIPR — FPI Exception Interrupt Priority Setting Register

This register is used to set the interrupt priority of FPI exception.

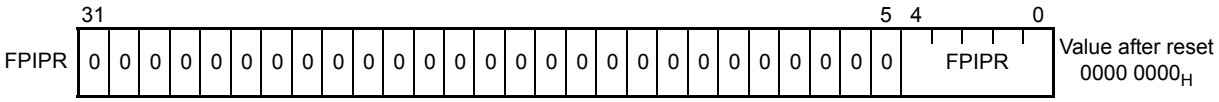



Table 3.31 FPIPR Register Contents

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 5 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 4 to 0 | FPIPR | These bits are used to specify the interrupt priority of floating-point operation exceptions (imprecise) (FPI). Specify values from 0 to 16.<br>Specifying 17 or greater is prohibited.<br>FPI exceptions are handled using the specified interrupt priority. If an FPI exception occurs at the same time as an interrupt that has the same priority, the FPI exception is prioritized.<br>**NOTE**<br>A set value of more than 16 is treated as 16. | R/W | 0 |

(b) ISPR — Priority of Interrupt being Serviced Register

This register retains the priority of the EIINTn interrupt being serviced by the CPU. This priority value is then used to perform priority ceiling processing when multiple interrupts occur.

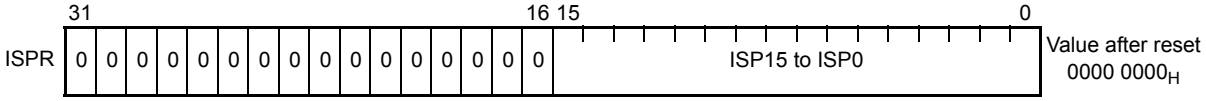



**Table 3.32 ISPR Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 16 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 15 to 0 | ISP15 to ISP0 | These bits indicate the acknowledgment status of an EIINTn interrupt with a priority that corresponds to the relevant bit position.<br>0: An interrupt request for an interrupt whose priority corresponds to the relevant bit position has not been acknowledged.<br>1: An interrupt request for an interrupt whose priority corresponds to the relevant bit position is being serviced by the CPU core.<br><br>The bit positions correspond to the following priority levels.<br><br>**Bit / Priority**<br>0 — Priority 0 (highest)<br>1 — Priority 1<br>...<br>14 — Priority 14<br>15 — Priority 15 (lowest)<br><br>When an interrupt request (EIINTn) is acknowledged, the bit corresponding to the acknowledged interrupt request is automatically set to 1. If PSW.EP is 0 when the EIRET instruction is executed, the bit with the highest priority among the ISP15 to ISP0 bits that are set to 1 (0 is the highest priority) is cleared to 0.*1<br>While a bit in this register is set to 1, same or lower priority interrupts (EIINTn) and FPI exceptions*2 are masked. Priority level judgment is therefore not performed when the system is determining whether to acknowledge an exception, meaning that exceptions will not be acknowledged.<br>When performing software-based priority control using the PMR register, be sure to clear this register by using the INTCFG.ISPC bit. | R*3 | 0 |

Note 1. Interrupt acknowledgment and auto-updating of values when the EIRET instruction is executed are disabled by setting (1) to the INTCFG.ISPC bit. It is recommended to enable auto-updating of values, so in normal cases, the INTCFG.ISPC bit should be cleared to 0.

Note 2. Since FPI exceptions have the same level of priority as EIINTn interrupts, they are affected by interrupts in the same way as the ISPR. The priority of FPI exceptions is set by the FPIPR register.

Note 3. This is R or R/W, depending on the setting of the INTCFG.ISPC bit. It is recommended to use this register as a read-only (R) register.

### (c) PMR — Interrupt Priority Masking Register

This register is used to mask the specified interrupt priority.

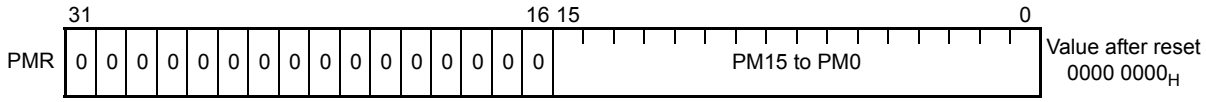



**Table 3.33 PMR Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 16 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 15 to 0 | PM15 to PM0 | These bits mask an interrupt request with a priority level that corresponds to the relevant bit position.<br>0: Servicing of an interrupt with a priority that corresponds to the relevant bit position is enabled.<br>1: Servicing of an interrupt with a priority that corresponds to the relevant bit position is disabled.<br><br>The bit positions correspond to the following priority levels:<br><br>**Bit** / **Priority**<br>0 / Priority 0 (highest)<br>1 / Priority 1<br>... <br>14 / Priority 14<br>15 / Priority 15 and priority 16 (lowest)<br><br>While a bit in this register is set to 1, interrupts (EIINTn) and FPI exceptions*1 with the priority corresponding to that bit are masked. Priority level judgment is therefore not performed when the system is determining whether to acknowledge an exception, meaning that exceptions will not be acknowledged*2. | R/W | 0 |

Note 1. Since FPI exceptions are specified as the same level of priority as that of interrupts (EIINTn), it is affected by the PMR like interrupts. The priority of FPI exceptions is set by the FPIPR register.

Note 2. Specify the masks by setting the bits to 1 in order from the lowest-priority bit. For example, $FF00_H$ can be set, but $F0F0_H$ or $00FF_H$ cannot.

### (d) ICSR — Interrupt Control Status Register

This register indicates the interrupt control status in the CPU.

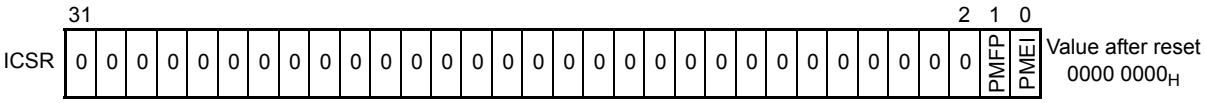



**Table 3.34 ICSR Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 2 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 1 | PMFP | This bit indicates that an FPI exception with the priority level masked by the PMR register exists. | R | 0 |
| 0 | PMEI | This bit indicates that an interrupt (EIINTn) with the priority level masked by the PMR register exists. | R | 0 |

(e) INTCFG — Interrupt Function Setting Register

This register is used to specify settings related to the CPU's internal interrupt function.

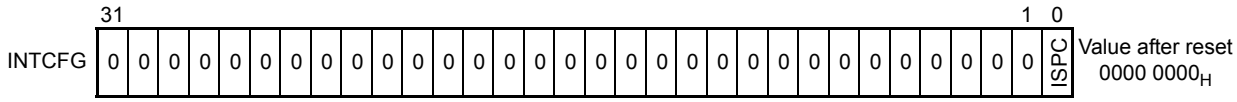



**Table 3.35 INTCFG Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 1 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 0 | ISPC | This bit specifies how the ISPR register is updated.<br>0: The ISPR register is automatically updated. Updates triggered by the program (via execution of LDSR instruction) are ignored.<br>1: The ISPR register is not automatically updated. Updates triggered by the program (via execution of LDSR instruction) are performed.<br><br>If this bit is cleared to 0, the bits of the ISPR register are automatically set to 1 when an interrupt (EIINTn) is acknowledged, and cleared to 0 when the EIRET instruction is executed. In this case, updating by the program (via execution of an LDSR instruction) is ignored.<br>If this bit is set to 1, the bits of the ISPR register are not updated by the acknowledgement of an interrupt (EIINTn) or by execution of the EIRET instruction. In this case, the bits can be updated by an LDSR instruction executed by the program.<br>In normal cases, the ISPC bit should be cleared to 0. When performing software-based control of interrupt priorities, however, set this bit (1) and perform priority control by using the PMR register. | R/W | 0 |

### (4) FPU Function Registers

The FPU uses the CPU general-purpose registers (r0 to r31). There are no register files used only for floating-point operations. The RH850/F1K supports single-precision floating-point instruction and thirty-two 32-bit registers can be specified.

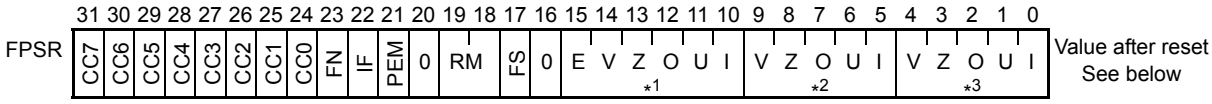
These registers correspond to general-purpose registers r0 to r31. The FPU can use the following system registers to control floating-point operation.

**Table 3.36 FPU System Registers**

| Register No. (regID, selID) | Symbol | Function | Access Permission |
|---|---|---|---|
| SR6, 0 | FPSR | Floating-point operation setting/status | CU and SV |
| SR7, 0 | FPEPC | Floating-point operation exception program counter | CU and SV |
| SR8, 0 | FPST | Floating-point status | CU |
| SR9, 0 | FPCC | Floating-point operation comparison result | CU |
| SR10, 0 | FPCFG | Floating-point function setting | CU |
| SR11, 0 | FPEC | Floating-point operation exception control | CU and SV |

#### (a) FPSR — Floating-point Operation Setting/Status Register

This register indicates the execution status of floating-point operations and any exceptions that occur.




Note 1. Cause bits (XC)
Note 2. Enable bits(XE)
Note 3. Preservation bits (XP)

**Table 3.37 FPSR Register Contents (1/2)**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 24 | CC[7:0] | These are the CC (condition) bits. They store the results of floating-point comparison instructions. The CC7 to CC0 bits are not affected by any instructions except the comparison instruction and LDSR instruction.<br>0: Comparison result is false<br>1: Comparison result is true | R/W | Undefined |
| 23 | FN | This bit enables flush-to-nearest mode. When the FN bit is set to 1, if the rounding mode is RN and the operation result is a subnormal number, the number is flushed to the nearest number. | R/W | 0 |
| 22 | IF | This bit accumulates and indicates information about the flushing of input operands. | R/W | 0 |
| 21 | PEM | This bit specifies whether to handle an exception as a precise exception. If the PEM bit is 1, exceptions that are caused by the execution of a floating-point operation instruction are handled as precise exceptions. | R/W | 0 |
| 20 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |

Table 3.37 FPSR Register Contents (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th><th>R/W</th><th>Value after Reset</th></tr>
<tr><td>19, 18</td><td>RM</td><td>These are the rounding mode control bits. The RM bits define the rounding mode that the FPU uses for all floating-point instructions.
<table>
<tr><th colspan="2">RM bits</th><th rowspan="2">Mnemonic</th><th rowspan="2">Description</th></tr>
<tr><th>19</th><th>18</th></tr>
<tr><td>0</td><td>0</td><td>RN</td><td>Rounds the result to the nearest representable value. If the value is exactly in-between the two nearest representable values, the result is rounded toward the value whose least significant bit is 0.</td></tr>
<tr><td>0</td><td>1</td><td>RZ</td><td>Rounds the result toward 0. The result is the nearest to the value that does not exceed the absolute value of the result with infinite accuracy.</td></tr>
<tr><td>1</td><td>0</td><td>RP</td><td>Rounds the result toward +∞. The result is nearest to a value greater than the accurate result with infinite accuracy.</td></tr>
<tr><td>1</td><td>1</td><td>RM</td><td>Rounds the result toward –∞. The result is nearest to a value less than the accurate result with infinite accuracy.</td></tr>
</table></td><td>R/W</td><td>00</td></tr>
<tr><td>17</td><td>FS</td><td>This bit enables values that cannot be normalized (subnormal numbers) to be flushed. If the FS bit is set, input operands and operation results that are subnormal numbers are flushed without causing an unimplemented operation exception (E). An input operand that is a subnormal number is flushed to 0 with the same sign.<br>Operation results that are subnormal numbers either become 0 or the minimum normalized number, depending on the rounding mode.
<table>
<tr><th rowspan="2">Operation Result that is a Subnormal Number</th><th colspan="4">Rounding Mode and Value after Flushing</th></tr>
<tr><th>RN*1</th><th>RZ</th><th>RP</th><th>RM</th></tr>
<tr><td>Positive</td><td>+0</td><td>+0</td><td>$+2^{Emin}$</td><td>+0</td></tr>
<tr><td>Negative</td><td>–0</td><td>–0</td><td>–0</td><td>$-2^{Emin}$</td></tr>
</table>
Note 1. If the rounding mode is RN and the FPSR.FN bit is set to 1, flushing will occur in the direction of higher accuracy.</td><td>R/W</td><td>1</td></tr>
<tr><td>16</td><td>—</td><td>(Reserved for future expansion. Be sure to set to 0.)</td><td>R</td><td>0</td></tr>
<tr><td>15 to 10</td><td>XC (E, V, Z, O, U, I)</td><td>These are the cause bits.</td><td>R/W</td><td>Undefined</td></tr>
<tr><td>9 to 5</td><td>XE (V, Z, O, U, I)</td><td>These are the enable bits.</td><td>R/W</td><td>0</td></tr>
<tr><td>4 to 0</td><td>XP (V, Z, O, U, I)</td><td>These are the preservation bits.</td><td>R/W</td><td>Undefined</td></tr>
</table>

(b) FPEPC — Floating-point Exception Program Counter Register

When an exception that is enabled by an enable bit occurs, the program counter (PC) of the instruction that caused the exception is stored.

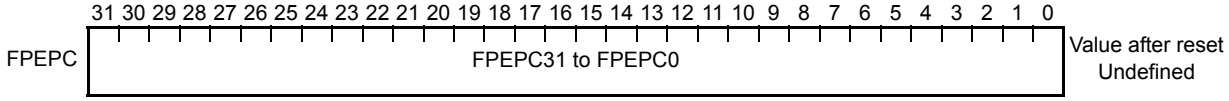
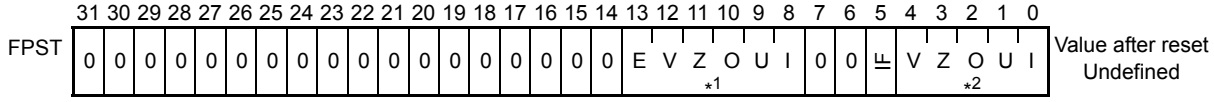



**Table 3.38 FPEPC Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 1 | FPEPC31 to FPEPC1 | These bits store the program counter (PC) of the floating-point instruction that caused the exception when a floating-point operation exception that is enabled by an enable bit occurs. | R/W | Undefined |
| 0 | FPEPC0 | This bit stores the program counter (PC) of the floating-point instruction that caused the exception when a floating-point operation exception that is enabled by an enable bit occurs.<br>Always set this bit to 0. | R | 0 |

(c) FPST — Floating-point Operation Status Register

This register reflects the contents of the FPSR register bits related to the operation status.




Note 1. Cause bits (XC)

Note 2. Preservation bits (XP)

**Table 3.39 FPST Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 14 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 13 to 8 | XC (E, V, Z, O, U, I) | These are cause bits. Values written to these bits are reflected in FPSR.XC bits. | R/W | Undefined |
| 7, 6 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 5 | IF | This bit accumulates and indicates information about the flushing of input operands. | R/W | 0 |
| 4 to 0 | XP (V, Z, O, U, I) | These are preservation bits. Values written to these bits are reflected in FPSR.XP bits. | R/W | Undefined |

### (d) FPCC — Floating-point Operation Comparison Result Register

This register reflects the contents of the FPSR.CC[7:0] bits.

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FPCC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CC7 | CC6 | CC5 | CC4 | CC3 | CC2 | CC1 | CC0 | Value after reset Undefined |

**Table 3.40 FPCC Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 8 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 7 to 0 | CC[7:0] | These are CC (condition) bits. They store the result of a floating-point comparison instruction. The CC[7:0] bits are not affected by any instructions except the comparison instruction and LDSR instruction. Values written to these bits are reflected in the CC[7:0] bits of FPSR.<br>0: Comparison result is false<br>1: Comparison result is true | R/W | Undefined |

### (e) FPCFG — Floating-point Operation Configuration Register

This register reflects the contents of the FPSR register bits related to the operation settings.

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FPCFG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RM | | 0 | 0 | 0 | V | Z | O *1 | U | I | Value after reset 0000 0000$_H$ |

Note 1. Enable bits (XE)

**Table 3.41 FPCFG Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 10 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 9, 8 | RM | These are rounding mode control bits. The RM bits define the rounding mode that the FPU uses for all floating-point instructions.<br>Values written to these bits are reflected in RM bits of FPSR. | R/W | 0 |

| RM bits | | | |
|---|---|---|---|
| 9 | 8 | Mnemonic | Description |
| 0 | 0 | RN | Rounds the result to the nearest representable value. If the value is exactly in-between the two nearest representable values, the result is rounded toward the value whose least significant bit is 0. |
| 0 | 1 | RZ | Rounds the result toward 0. The result is the nearest to the value that does not exceed the absolute value of the result with infinite accuracy. |
| 1 | 0 | RP | Rounds the result toward + ∞. The result is nearest to a value greater than the accurate result with infinite accuracy. |
| 1 | 1 | RM | Rounds the result toward – ∞. The result is nearest to a value less than the accurate result with infinite accuracy. |

| | | | | |
|---|---|---|---|---|
| 7 to 5 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 4 to 0 | XE (V, Z, O, U, I) | These are the enable bits. | R/W | 0 |

(f) FPEC — Floating-point Exception Control Register

This register controls the floating-point operation exception.

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FPEC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FPIVD | Value after reset 0000 0000$_H$ |

Table 3.42 FPEC Register Contents

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 1 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 0 | FPIVD*1 | This bit indicates the status of reporting the FPI exception.<br>If this bit is set to 1, the FPI exception is reported to the CPU but is not acknowledged. It is automatically cleared to 0 when the CPU acknowledges the FPI exception.<br>While this bit is set to 1, all the floating-point instructions are invalidated.<br>Report of the FPI exception can be canceled by clearing (0) this bit by the LDSR instruction while it is set to 1. When report of the FPI exception is canceled, the CPU does not acknowledge the FPI exception.<br>0: FPI exception is not reported.<br>1: FPI exception is reported. | R/W | 0 |

Note 1. The FPIVD bit can only be cleared to 0 by the write operation of the LDSR instruction. It cannot be set to 1.

### (5) MPU Function Registers

Table 3.43 MPU Function System Registers (1/2)

| Register No. (regID, selID) | Symbol | Function | Access Permission |
|---|---|---|---|
| SR0, 5 | MPM | Memory protection operation mode setting | SV |
| SR1, 5 | MPRC | MPU region control | SV |
| SR4, 5 | MPBRGN | MPU base region number | SV |
| SR5, 5 | MPTRGN | MPU end region number | SV |
| SR8, 5 | MCA | Memory protection setting check address | SV |
| SR9, 5 | MCS | Memory protection setting check size | SV |
| SR10, 5 | MCC | Memory protection setting check command | SV |
| SR11, 5 | MCR | Memory protection setting check result | SV |
| SR0, 6 | MPLA0 | Protection area lower limit address | SV |
| SR1, 6 | MPUA0 | Protection area upper limit address | SV |
| SR2, 6 | MPAT0 | Protection area attribute | SV |
| SR4, 6 | MPLA1 | Protection area lower limit address | SV |
| SR5, 6 | MPUA1 | Protection area upper limit address | SV |
| SR6, 6 | MPAT1 | Protection area attribute | SV |
| SR8, 6 | MPLA2 | Protection area lower limit address | SV |
| SR9, 6 | MPUA2 | Protection area upper limit address | SV |
| SR10, 6 | MPAT2 | Protection area attribute | SV |
| SR12, 6 | MPLA3 | Protection area lower limit address | SV |
| SR13, 6 | MPUA3 | Protection area upper limit address | SV |
| SR14, 6 | MPAT3 | Protection area attribute | SV |
| SR16, 6 | MPLA4 | Protection area lower limit address | SV |
| SR17, 6 | MPUA4 | Protection area upper limit address | SV |

Table 3.43 MPU Function System Registers (2/2)

| Register No. (regID, selID) | Symbol | Function | Access Permission |
|---|---|---|---|
| SR18, 6 | MPAT4 | Protection area attribute | SV |
| SR20, 6 | MPLA5 | Protection area lower limit address | SV |
| SR21, 6 | MPUA5 | Protection area upper limit address | SV |
| SR22, 6 | MPAT5 | Protection area attribute | SV |
| SR24, 6 | MPLA6 | Protection area lower limit address | SV |
| SR25, 6 | MPUA6 | Protection area upper limit address | SV |
| SR26, 6 | MPAT6 | Protection area attribute | SV |
| SR28, 6 | MPLA7 | Protection area lower limit address | SV |
| SR29, 6 | MPUA7 | Protection area upper limit address | SV |
| SR30, 6 | MPAT7 | Protection area attribute | SV |
| SR0, 7 | MPLA8 | Protection area lower limit address | SV |
| SR1, 7 | MPUA8 | Protection area upper limit address | SV |
| SR2, 7 | MPAT8 | Protection area attribute | SV |
| SR4, 7 | MPLA9 | Protection area lower limit address | SV |
| SR5, 7 | MPUA9 | Protection area upper limit address | SV |
| SR6, 7 | MPAT9 | Protection area attribute | SV |
| SR8, 7 | MPLA10 | Protection area lower limit address | SV |
| SR9, 7 | MPUA10 | Protection area upper limit address | SV |
| SR10, 7 | MPAT10 | Protection area attribute | SV |
| SR12, 7 | MPLA11 | Protection area lower limit address | SV |
| SR13, 7 | MPUA11 | Protection area upper limit address | SV |
| SR14, 7 | MPAT11 | Protection area attribute | SV |
| SR16, 7 | MPLA12 | Protection area lower limit address | SV |
| SR17, 7 | MPUA12 | Protection area upper limit address | SV |
| SR18, 7 | MPAT12 | Protection area attribute | SV |
| SR20, 7 | MPLA13 | Protection area lower limit address | SV |
| SR21, 7 | MPUA13 | Protection area upper limit address | SV |
| SR22, 7 | MPAT13 | Protection area attribute | SV |
| SR24, 7 | MPLA14 | Protection area lower limit address | SV |
| SR25, 7 | MPUA14 | Protection area upper limit address | SV |
| SR26, 7 | MPAT14 | Protection area attribute | SV |
| SR28, 7 | MPLA15 | Protection area lower limit address | SV |
| SR29, 7 | MPUA15 | Protection area upper limit address | SV |
| SR30, 7 | MPAT15 | Protection area attribute | SV |

### (a) MPM — Memory Protection Operation Mode Register

The memory protection mode register is used to define the basic operating state of the memory protection function. The settings for this register are normally specified once on startup and are not changed during program execution.

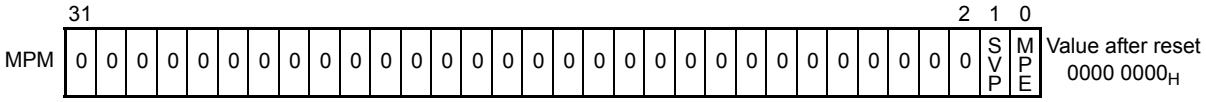



**Table 3.44 MPM Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 2 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 1 | SVP | In SV mode (when PSW.UM = 0), this bit is used to specify whether to restrict access according to the SX, SW, and SR bits of the MPAT register for each protection area.*1<br>0: As usual, implicitly enable all access in SV mode.<br>1: Restrict access according to the SX, SW, and SR bits even in SV mode.*2 | R/W | 0 |
| 0 | MPE | This bit is used to specify whether to enable or disable the MPU function.<br>0: Disable<br>1: Enable | R/W | 0 |

Note 1. If the SVP bit is set to 1, access will be restricted in accordance with the setting for each protection area, even in SV mode. Therefore, specify the protection area beforehand so that the access from the program which set the SVP bit is not restricted.

Note 2. If access is restricted in SV mode, execution of MDP exceptions or the MIP exception handling itself might not be possible depending on the settings. Be careful to specify settings so that access by the exception handler and to the memory area necessary for exception handling is permitted.

### (b) MPRC — MPU Region Control Register

Bits used to perform special memory protection function operations are located in this register.

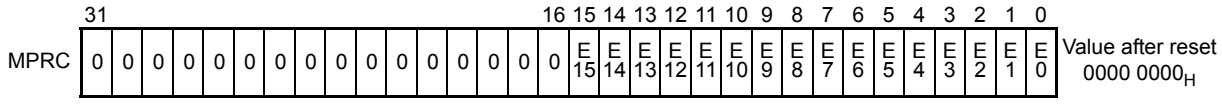



**Table 3.45 MPRC Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 16 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 15 to 0 | E15 to E0 | These are the enable bits for each protection area. Bit En is a copy of bit MPATn.E (where n = 15 to 0). | R/W | 0 |

(c) MPBRGN — MPU Base Region Register

This register indicates the minimum usable MPU area number.

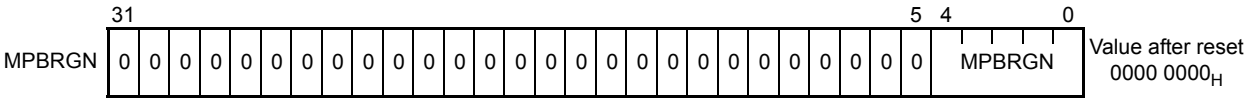



**Table 3.46 MPBRGN Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 5 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 4 to 0 | MPBRGN | These bits indicate the smallest number of an MPU area.<br>These bits always indicate 0. | R | 0 |

(d) MPTRGN — MPU End Region Register

This register indicates the maximum usable MPU area number + 1.

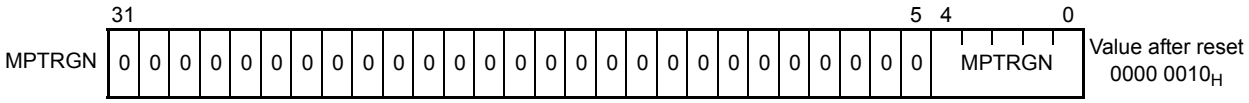



**Table 3.47 MPTRGN Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 5 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 4 to 0 | MPTRGN | These bits indicate the largest number of an MPU area + 1.<br>These bits indicate the maximum number of MPU areas incorporated into the hardware. | R | 10000B |

(e) MCA — Memory Protection Setting Check Address Register

This register is used to specify the base address of the area for which a memory protection setting check is to be performed.

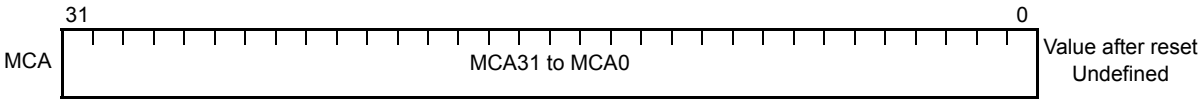



**Table 3.48 MCA Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 0 | MCA31 to MCA0 | These bits are used to specify the start address of the memory area that is subject to a memory protection setting check in bytes. | R/W | Undefined |

### (f) MCS — Memory Protection Setting Check Size Register

This register is used to specify the size of the area for which a memory protection setting check is to be performed.

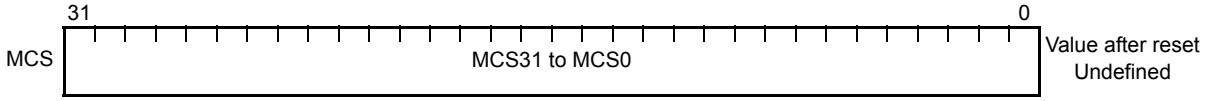



**Table 3.49 MCS Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 0 | MCS31 to MCS0 | These bits are used to specify the size of the memory area that is subject to a memory protection setting check and the size of the target area in bytes.<br>Because the specified size is assumed to represent an unsigned integer, it is not possible to check an area in the direction in which the address value decreases relative to the MCA register value.<br>Do not specify 0000 $0000_H$ for the MCS register. | R/W | Undefined |

### (g) MCC — Memory Protection Setting Check Command Register

This register is used to specify the base address of the area where memory protection settings are checked.

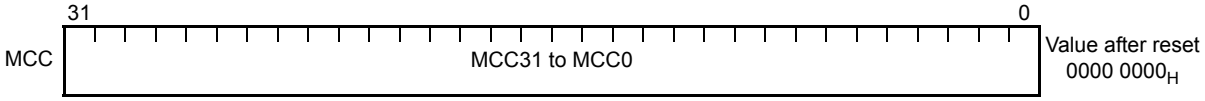



**Table 3.50 MCC Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 0 | MCC31 to MCC0 | When any value is written to the MCC register, a memory protection setting check starts. By setting up the MCA/MCS register and then writing to the MCC register, results are stored in MCR.<br>Because the check is started by any written value, a check can be started by using r0 as the source register without using any unnecessary registers.<br>Note that, for the check, the results are applied according to each area setting regardless of the state of the PSW.UM bit.<br>When the MCC register is read, value 0000 $0000_H$ is always returned. | R/W | 0 |

(h) MCR — Memory Protection Setting Check Result Register

This register is used to store the results of a memory protection setting check.

Be sure to clear bits 31 to 9, 7 and 6.

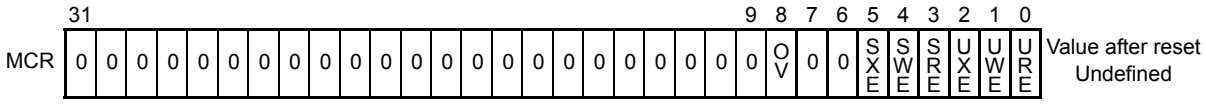



**Table 3.51 MCR Register Contents**

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 9 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 8 | OV | If the specified area includes 0000 0000$_H$ or 7FFF FFFF$_H$, 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |
| 7, 6 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 5 | SXE | If the specified area is contained within one of the protection areas and execution is permitted for that area in supervisor mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |
| 4 | SWE | If the specified area is contained within one of the protection areas and writing to that area is permitted in supervisor mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |
| 3 | SRE | If the specified area is contained within one of the protection areas and reading from that area is permitted in supervisor mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |
| 2 | UXE | If the specified area is contained within one of the protection areas and execution is permitted for that area in user mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |
| 1 | UWE | If the specified area is contained within one of the protection areas and writing to that area is permitted in user mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |
| 0 | URE | If the specified area is contained within one of the protection areas and reading from that area is permitted in user mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |

(i) MPLAn — Protection Area Lower Limit Address Register

These registers indicate the lower limit address of area n (where n = 0 to 15).

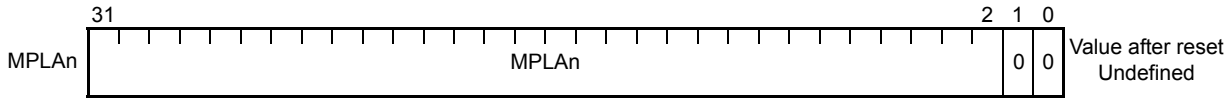



Table 3.52 MPLAn Register Contents

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 2 | MPLA31 to MPLA2 | These bits indicate the lower limit address of area n.<br>For MPLA1 and MPLA0, 0 is used implicitly. | R/W | Undefined |
| 1, 0 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |

(j) MPUAn — Protection Area Upper Limit Address Register

These registers indicate the upper limit address of area n (where n = 0 to 15).

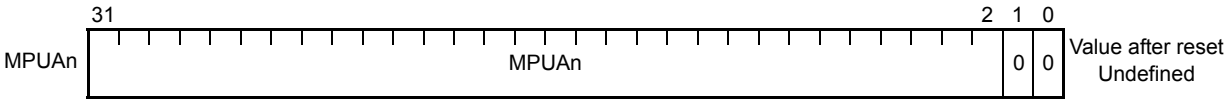



Table 3.53 MPUAn Register Contents

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 2 | MPUA31 to MPUA2 | These bits indicate the upper limit address of area n.<br>For MPUAn.MPUA1 and MPUA0, 1 is used implicitly. | R/W | Undefined |
| 1, 0 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |

(k) MPATn — Protection Area Attribute Register

These registers indicate the attributes of area n (where n = 0 to 15).

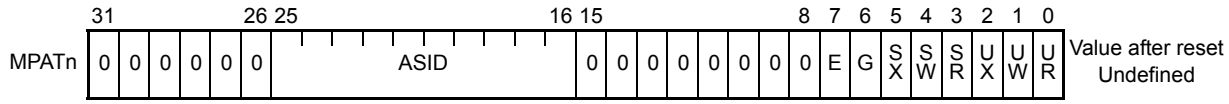



Table 3.54 MPATn Register Contents (1/2)

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 26 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 25 to 16 | ASID | These bits indicate the ASID value to be used as the area match condition. | R/W | Undefined |
| 15 to 8 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 7 | E | This bit indicates whether area n is enabled or disabled.<br>0: Area n is disabled.<br>1: Area n is enabled. | R/W | 0 |
| 6 | G | 0: Areas match only if ASIDs are equal.<br>1: Areas match even if ASIDs are not equal.<br>If this bit is 0, MPATn.ASID = ASID.ASID is used as the area match condition.<br>If this bit is 1, areas may match even if the values of MPATn.ASID and ASID.ASID are not equal. | R/W | Undefined |

Table 3.54 MPATn Register Contents (2/2)

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 5 | SX | This bit indicates the execution privilege for the supervisor mode.*1<br>0: Execution is disabled.<br>1: Execution is enabled. | R/W | Undefined |
| 4 | SW | This bit indicates whether writing is enabled in the supervisor mode.*1<br>0: Writing is disabled.<br>1: Writing is enabled. | R/W | Undefined |
| 3 | SR | This bit indicates whether writing is enabled in the supervisor mode.*1<br>0: Reading is disabled.<br>1: Reading is enabled. | R/W | Undefined |
| 2 | UX | This bit indicates the execution privilege for the user mode.<br>0: Execution is disabled.<br>1: Execution is enabled. | R/W | Undefined |
| 1 | UW | This bit indicates whether writing is enabled in the user mode.<br>0: Writing is disabled.<br>1: Writing is enabled. | R/W | Undefined |
| 0 | UR | This bit indicates whether writing is enabled in the user mode.<br>0: Reading is disabled.<br>1: Reading is enabled. | R/W | Undefined |

Note 1. If access is restricted in SV mode, execution of MDP exceptions or the MIP exception handling itself might not be possible depending on the settings. Be careful to specify settings so that access by the exception handler and to the memory area necessary for exception handling is permitted.

### (6) Cache Operation Function Registers

The RH850/F1K does not include a cache operation function, so all the following registers return a value of 0 when read, and writing to these registers is ignored.

**Table 3.55 Cache Operation Function Registers**

| Register No. (regID, selID) | Symbol | Function | Access Permission |
|---|---|---|---|
| SR12, 4 | BWERRL | Not implemented. A value of 0 is returned when read and writing is ignored. | SV |
| SR13, 4 | BWERRH | | SV |
| SR14, 4 | BRERRL | | SV |
| SR15, 4 | BRERRH | | SV |
| SR16, 4 | ICTAGL | | SV |
| SR17, 4 | ICTAGH | | SV |
| SR18, 4 | ICDATL | | SV |
| SR19, 4 | ICDATH | | SV |
| SR20, 4 | DCTAGL | | SV |
| SR21, 4 | DCTAGH | | SV |
| SR22, 4 | DCDATL | | SV |
| SR23, 4 | DCDATH | | SV |
| SR24, 4 | ICCTRL | | SV |
| SR25, 4 | DCCTRL | | SV |
| SR26, 4 | ICCFG | | SV |
| SR27, 4 | DCCFG | | SV |
| SR28, 4 | ICERR | | SV |
| SR29, 4 | DCERR | | SV |

## 3.2.1.3 Instruction

See *"Instruction" in Software Manual*.

A snooze instruction halts operation of the CPU core for 32 clocks.

## 3.2.2 Buffers for Code Flash

### 3.2.2.1 Features

CPU accesses Code flash by two paths; instruction fetch access is direct to Flash interface, and data access is via System interconnect to Code Flash. Both paths equip buffers, which can be cleared by software. See also **Figure 33.1 "Block Diagram of Code Flash ECC"** for ECC decoders in these paths.

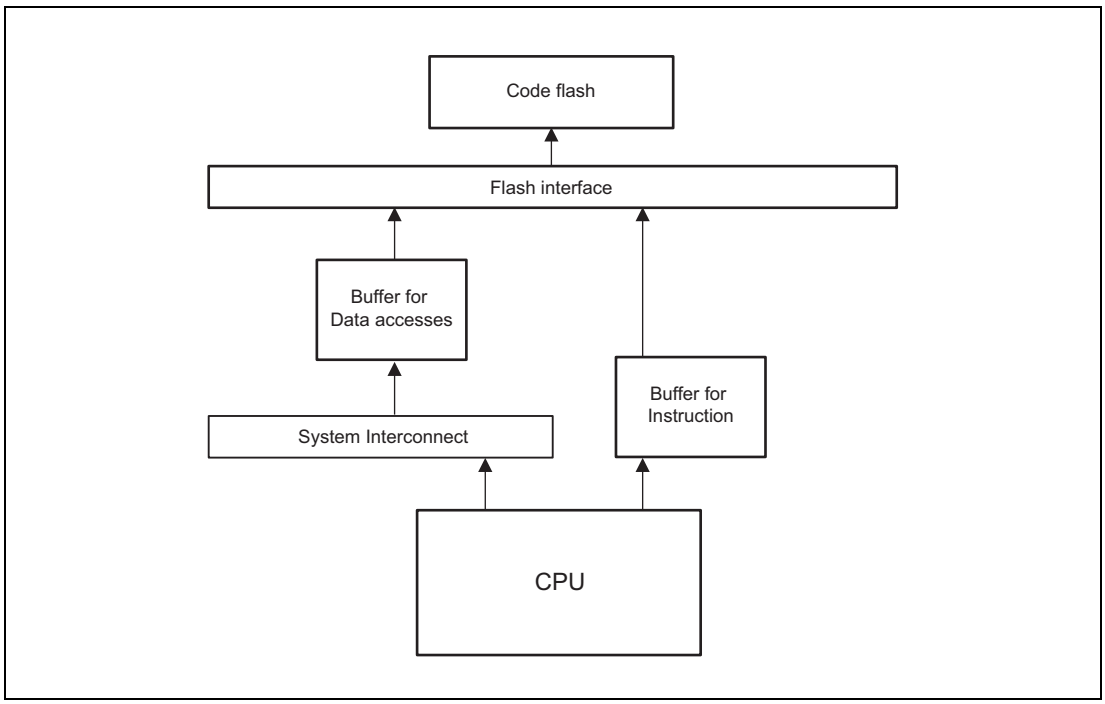



**Figure 3.2 Buffers for Code Flash**

### 3.2.2.2 Function of Buffers

One-line buffer with 128 bits is mounted for instruction fetches to code flash. The data is read out from the buffer after the next access to the same address, so the code flash is not accessed again within 128 bits location.

One-line buffer with 128 bits is mounted as a data buffer. This buffer is not only used by the CPU but also used by DMA via system interconnect. The data is read out from the buffer if the next access is within the same 128 bits boundary.

One buffer control register named FBUFCCTL is equipped. Using the FBUFCCTL register, the software can clear these two buffers.

### 3.2.2.3 Registers for Buffer Control

#### (1) List of Buffer Control Registers

Table 3.56 Buffer Control Register (Base Address: FFC5 $B000_H$)

| Module Name | Address Offset | Size (Byte) | Register Name | Abbreviation | Right | R/W | Operable Bit | | | | Value after Reset |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | 1 | 8 | 16 | 32 | |
| FBUF_CTRL | $+000_H$ | 4 | Flash Buffer Clear Control Register | FBUFCCTL | — | R/W | — | √ | √ | √ | 0000 $0000_H$ |

#### (2) Register Sets

| | 31 | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FBUFCCTL | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FBUFCLR |

Value after reset
0000 $0000_H$

Table 3.57 FBUFCCTL Register Contents

| Bit Position | Bit Name | Function | R/W | Value after Reset |
|---|---|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. | R | 0 |
| 0 | FBUFCLR | Buffer clear bits. To clear buffers, write 1 to this bit and then write 0.<br>0: Buffers are valid<br>1: Buffers are invalid (cleared) | R/W | 0 |

Please do following procedures when you want to clear the buffers.

Step 1: Write 0 to FBUFCCTL.FBUFCLR

Step 2: Write 1 to FBUFCCTL.FBUFCLR

Step 3: Write 0 to FBUFCCTL.FBUFCLR

Step 4: Read the FBUFCCTL register (dummy read)

Step 5: Execute the SYNCP instruction

Step 6: Execute the SYNCI instruction

If you do not do Step 3 after Step 2, the buffers are kept invalid during FBUFCCTL.FBUFCLR = 1.

## 3.2.3 Reliability Functions

### 3.2.3.1 PE Guard Function (PEG)

#### (1) Overview of the PEG Function

The PEG is a constituent of the slave guard system to prevent unauthorized access to the resources in the CPU (PE) from an external master. This function protects access to the local RAM and the retention RAM in the PE. In the initial state after a reset, access by masters other than own PE is disabled. Setting the registers listed in **(3) List of PEG Protection Setting Registers** enables access by masters other than own PE.

(1) Detecting PE guard violation
If an external master makes an unauthorized access to the resource area in a PE for which PE guard is set, the access is detected as a PE guard violation.

(2) Blocking unauthorized access
When a PE guard violation is detected, unauthorized access to the internal resources of the PE are blocked to prevent unauthorized modification of the contents of PE resources.

(3) Notifying occurrence of violation
An error response to an unauthorized access is sent to the request source of external master. When DMA Controller makes an unauthorized access, meanwhile, a DMA transfer error is detected. A PE guard violation is notified as INTGUARD interrupt request which is a source of FEINT.

#### (2) Protection Made by SPID

- Setting PEG Protection
  - Up to four areas can be set depending on the Local RAM address and the retention RAM address of the own PE.
  - The area range is specified by the base address and the mask bit (4 kbytes to 4 Gbytes).
  - “Read enable” and “write enable” can be set for each area.
  - “Enable” or “disable” can be selected based on the system protection identifier (SPID) for each area.
- Procedure for permitting access by using the system protection identifier (SPID)

1. Is the area subject to access is the local RAM area or the retention RAM area? If so, go to step 2.
2. Is the area subject to access is within the range of valid areas 0, 1, 2, or 3? If so, go to step 3. Otherwise, return an error response.
3. Are all the conditions below for the relevant area satisfied? If so, permit access.
   - The system protection identifier (SPID) is enabled.
   - Required operations (read/write) are enabled.

   Otherwise, return an error response.

### (3) List of PEG Protection Setting Registers

Specify the necessary settings in the registers below to protect PE resources from unauthorized access by an external master.

- Whether to permit access to the local RAM and the retention RAM in the PE can be specified.

**Table 3.58 PEG registers (Base Address : FFFE E600$_H$)**

| Module Name | Address Offset | Size (Byte) | Register Name | Abbreviation | Right | R/W | Operable Bit | | | | Value after Reset |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | 1 | 8 | 16 | 32 | |
| PEG | +00C$_H$ | 4 | PEG SPID control register | PEGSP | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| | +080$_H$ | 4 | PEG area 0 mask setting register | PEGG0MK | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| | +084$_H$ | 4 | PEG area 0 base setting register | PEGG0BA | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| | +090$_H$ | 4 | PEG area 1 mask setting register | PEGG1MK | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| | +094$_H$ | 4 | PEG area 1 base setting register | PEGG1BA | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| | +0A0$_H$ | 4 | PEG area 2 mask setting register | PEGG2MK | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| | +0A4$_H$ | 4 | PEG area 2 base setting register | PEGG2BA | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| | +0B0$_H$ | 4 | PEG area 3 mask setting register | PEGG3MK | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| | +0B4$_H$ | 4 | PEG area 3 base setting register | PEGG3BA | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |

### (4) Register Set

#### (a) PEGSP — PEG SPID Control Register

**Access:** PEGSP register can be read or written in 32-bit units.
PEGSPL register can be read or written in 16-bit units.
PEGSPLL register can be read or written in 8-bit units.

**Address:** PEGSP: FFFE E60C$_H$
PEGSPL: FFFE E60C$_H$
PEGSPLL: FFFE E60C$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SPEN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 3.59 PEGSP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | SPEN | Access permission to external master with specified SPID.<br>0: Not permit.<br>1: Permit. |

### (b) PEGGnMK — PEG Area n Mask Setting Register (n = 0 to 3)

The PEGGnMK register defines which bits of PEGGnBA.GnBASE are compared with the access address. If bit PEGGnMK.GnMASK[m] is cleared, bit PEGGnBA.GnBASE[m] is compared with bit m of the access address.

**Access:** PEGGnMK register can be read or written in 32-bit units.
PEGGnMKL, PEGGnMKH registers can be read or written in 16-bit units.
PEGGnMKLH, PEGGnMKHL, PEGGnMKHH registers can be read or written in 8-bit units.

**Address:** PEGGnMK: FFFE $E680_H$ + ($10_H$ × n)
PEGGnMKL: FFFE $E680_H$ + ($10_H$ × n),
PEGGnMKH: FFFE $E682_H$ + ($10_H$ × n)
PEGGnMKLH: FFFE $E681_H$ + ($10_H$ × n),
PEGGnMKHL: FFFE $E682_H$ + ($10_H$ × n),
PEGGnMKHH: FFFE $E683_H$ + ($10_H$ × n)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GnMASK | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GnMASK | | | | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 3.60 PEGGnMK Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 12 | GnMASK | 0: Target bits are compared when determining the PE guard area.<br>1: Target bits are not compared when determining the PE guard area. |
| 11 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

**NOTE**

When you write to the PEGGnMK register, the corresponding GnEN bit in the PEGGnBA register is cleared automatically.

### (c) PEGGnBA — PEG Area n Base Setting Register (n = 0 to 3)

In combination with the PEGGnMK register, this register specifies a range or ranges within PE guard protection area n. Setting the GnEN bit to 1 validates the access enable conditions specified by this register and the PEGGnMK register.

**Access:** PEGGnBA register can be read or written in 32-bit units.
PEGGnBAL, PEGGnBAH registers can be read or written in 16-bit units.
PEGGnBALL, PEGGnBALH, PEGGnBAHL, PEGGnBAHH registers can be read or written in 8-bit units.

**Address:** PEGGnBA: FFFE E684$_H$ + (10$_H$ × n)
PEGGnBAL: FFFE E684$_H$ + (10$_H$ × n),
PEGGnBAH: FFFE E686$_H$ + (10$_H$ × n)
PEGGnBALL: FFFE E684$_H$ + (10$_H$ × n),
PEGGnBALH: FFFE E685$_H$ + (10$_H$ × n),
PEGGnBAHL: FFFE E686$_H$ + (10$_H$ × n),
PEGGnBAHH: FFFE E687$_H$ + (10$_H$ × n)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GnBASE | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GnBASE | | | | — | — | — | — | GnSP3 | GnSP2 | GnSP1 | GnSP0 | — | GnWR | GnRD | GnEN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W |

**Table 3.61 PEGGnBA Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 12 | GnBASE | Base address that specifies the range of PE guard protection area n. |
| 11 to 8 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 7 | GnSP3 | Access permission setting from SPID = 3 external master to PE guard protection area n.<br>0: Not permit.<br>1: Permit. |
| 6 | GnSP2 | Access permission setting from SPID = 2 external master to PE guard protection area n.<br>0: Not permit.<br>1: Permit. |
| 5 | GnSP1 | Access permission setting from SPID = 1 external master to PE guard protection area n.<br>0: Not permit.<br>1: Permit. |
| 4 | GnSP0 | Access permission setting from SPID = 0 external master to PE guard protection area n.<br>0: Not permit.<br>1: Permit. |
| 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 | GnWR | Write access permission to PE guard protection area n.<br>0: Write access is disabled.<br>1: Write access is enabled. |
| 1 | GnRD | Read access permission to PE guard protection area n.<br>0: Read access is disabled.<br>1: Read access is enabled. |

Table 3.61 PEGGnBA Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | GnEN | Enables or Disables the setting for the access enable conditions to PE guard protection area n.<br>0: Settings for access enable conditions are disabled.<br>1: Settings for access enable conditions are enabled. |

**NOTE**

When you write to the PEGGnMK register, the corresponding GnEN bit in the PEGGnBA register is cleared automatically.

### 3.2.3.2 System Error Generator Function (SEG)

SEG (SysErrGen) controls interrupt notification and recording after a system error occurred by a data access.

Multiple error occurrence inputs are categorized according to error factor, and are processed sequentially from the highest-priority error factor, generating an FE-level asynchronous exception (SYSERR).

The bit position of the SEGFLAG register becomes the error factor priority. Error factors of lower bits take precedence over error factors of upper bits.

Error address information is recorded only once regardless of error frequency.

The error with the highest priority among the error factors is valid when errors occur simultaneously. Recorded error address information is not overwritten by subsequent errors.

#### (1) List of SEG Function Control Registers

Table 3.62 SEG Registers (Base Address: FFFE E980$_H$)

| Module Name | Address Offset | Size (Byte) | Register Name | Abbreviation | Right *1 | R/W | Operable Bit | | | | Value after Reset |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | 1 | 8 | 16 | 32 | |
| SEG | +00$_H$ | 2 | SEG Error control register | SEGCONT | SV | R/W | — | — | √ | — | 0000$_H$ |
| | +02$_H$ | 2 | SEG Error flag register | SEGFLAG | SV | R/W | — | — | √ | — | 0000$_H$ |
| | +08$_H$ | 4 | SEG Error address information register | SEGADDR | SV | R/W | — | — | — | √ | Undefined (retained) |

Note 1. Registers for which “SV” is described are writable with the SV right (UM = 0). Attempting to write, if these conditions do not hold, leads to a SYSERR exception with setting VCIF flag. No restriction is provided for read accesses.

### (2) Register Set

#### (a) SEGCONT — SEG Error Control Register

This register is used to enable (= 1) or disable (= 0) notification of SysErr request in response to error flags that store the error occurrence status for each factor.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | VCSE | APIE | — | — | TCME | — | VCIE | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R | R | R/W | R | R/W | R | R | R | R |

**Table 3.63 SEGCONT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 11 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 10 | VCSE | This bit enables notification of an error response detected inside system interconnect. |
| 9 | APIE | This bit enables notification of an error response from peripherals. The error notification includes the following cases:<br>• Error response from peripherals in write access<br>• PBG error in write access |
| 8 to 7 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 6 | TCME | This bit enables notification of an error during data access to its own local RAM and retention RAM.<br>The error notification includes the following cases:<br>• ECC uncorrectable error<br>• Detection of an access to RAM unimplemented area |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 | VCIE | • This bit enables notification of an error response during access to P-Bus. The error notification includes the following cases:<br>– Error response from peripherals (excluding errors in write access)<br>– PBG error (excluding errors in write access)<br>• This bit enables notification of an error response during access to code flash. The error notification includes the following cases:<br>– ECC uncorrectable error<br>• This bit enables notification of an error response when accessing to a part of access prohibited areas in address map. |
| 3 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

(b) SEGFLAG — SEG Error Flag Register

This register indicates error flags that store error occurrence status of each factors. The flags are set to 1 by an error occurrence input. The flags are not automatically cleared to 0. Both setting and clearing of each flag are supported in writing to the register.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | VCSF | APIF | — | — | TCMF | — | VCIF | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R | R | R/W | R | R/W | R | R | R | R |

**Table 3.64 SEGFLAG Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 11 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 10 | VCSF | Flag corresponding to bit 10 of the SEGCONT register |
| 9 | APIF | Flag corresponding to bit 9 of the SEGCONT register |
| 8 to 7 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 6 | TCMF | Flag corresponding to bit 6 of the SEGCONT register |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 | VCIF | Flag corresponding to bit 4 of the SEGCONT register |
| 3 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

**NOTE**

An error may lead to setting of multiple error flags in SEG.

### (c) SEGADDR — Error Address Information Register

Address information (one record) which is notified with error occurence is stored in the register. The register is not updated while one or more bits in SEGFLAG register are set.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Address[31:16] | | | | | | | | | | | | | | | |
| Value after reset | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Address[15:0] | | | | | | | | | | | | | | | |
| Value after reset | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

×: Undefined (retained)

**Table 3.65 SEGADDR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | Address | These bits store the error address information. |

Note 1. SEGADDR stores error address information in case of an error occurrence related to VCIF bit or TCMF bit in SEGFLAG register. SEGADDR register stores all 0 data in case of an error occurrence related to VCSF bit and APIF bit in SEGFLAG register.

Note 2. In case of an error occurrence related to TCMF bit in SEGFLAG register, bit[18:0] of the error address are stored in SEGADDR[18:0] and SEGADDR[31:19] are filled with 0.

### (3) SEG Function

#### (a) SEG Function: SYSERR request notification by error Flag

- Setting an error flag takes precedence over clearing the same flag.
  - A simultaneous clearing operation is ignored.
- Priority of error factors
  - The bit position of each flags in SEGFLAG register which error notification is enabled by SEGCONT register becomes the error factor priority. Error factors of lower bits take precedence over error factors of upper bits. Notification is made from the highest-priority error factor.
  - The bit position of error factors is reported as a “SysErr factor code.”
- Conditions for starting SysErr request notification
  - Even if a flag which error notification is disabled by SEGCONT register is set to 1, notification is not made.
  - Notification is made immediately after a flag which error notification is enabled by SEGCONT register is set to 1.
  - After clearing of a flag, notification is made if an other flag which error notification is enabled by SEGCONT register remains set.
- Finishing notification at a SysErr acknowledgement
  - Even after notification is finished, the flag is not cleared automatically.
  - Notification is not made until setting or clearing the flag again.
  - If an error flag that is prioritized higher than the error factor is set prior to an acknowledgement, the notification information may be replaced with a higher prioritized SysErr factor code.

#### (b) SEG Function: Recording Error Address Information

- When an error which error notification is enabled by SEGCONT register occurs, the error address is retained in the SEGADDR register.
  - No information is retained by setting or clearing an error flag in SEGFLAG register.
  - When multiple error occurrence inputs are present simultaneously, information other than the prioritized error factor is not retained.
- While a flag which error notification is enabled by SEGCONT register is set to 1, overwrite to the SEGADDR register is inhibited.
  - If error occurrence input continues, information of subsequent error factors is not retained.
  - To reset the inhibition of overwrite to the register, clear either SEGCONT or SEGFLAG register (or both of them).

#### (c) Supplementary Notes on SYSERR Exception

- Even when a SYSERR exception occurs, the value of the PSW.EBV bit is held, and the base address of the exception handler does not switch.

## 3.3 Notes

### 3.3.1 Synchronization of Store Instruction Completion and Subsequent Instruction Execution

When a control register is updated by a store instruction, there is a time lag after the CPU executes the store instruction and before the control register is actually updated. Therefore, if the updated content of the control register is to be used by the instruction following the store instruction, the appropriate synchronization is required. How to perform synchronization processing is shown below.

For the procedures to synchronize updating system registers by LDSR instruction and the subsequent instruction execution, see *APPENDIX A. Hazard Resolution Procedure for System Registers* in the *RH850G3KH User's Manual: Software.*

**When the updated results in the control registers are to be used by the subsequent instruction:**

Example 1: An interrupt is enabled by execution of an EI instruction after an interrupt request is cleared by access from the control register in the INTC2 and the peripheral circuits. Proceed as follows in this case.

(1) Execute the store instruction to update a control register (ST.W, etc).

(2) Perform a dummy read of the above control register (LD.W, etc).

(3) Execute SYNCP.

(4) Execute the subsequent instruction (EI).

In case of RH850/F1K, SYNCM instruction has the same effects as above-mentioned (2) and (3).

Example 2: When you must wait until a control register (control register A) has been completely updated before accessing another control register (control register B), execute similar processing. For example, different peripheral functions are linked, or the interrupt mask for INTC is cleared after the peripheral function is set. Note that this processing is not required if the control registers A and B belong to the same peripheral group.

(1) Execute the store instruction to update the control register A (ST.W, etc).

(2) Perform a dummy read of the above control register (LD.W, etc).

(3) Execute SYNCP.

(4) Execute the store instruction to access the control register B (ST.W, LD.W, etc).

In case of RH850/F1K, SYNCM instruction has the same effects as above-mentioned (2) and (3).

The similar processing is also required when starting to access a memory or control register to be protected is started after a safety function (such as some kind of memory protection and ECC) has been completely set up.

**When the updated results of the control register or memory to be used in the instruction fetch of the subsequent instruction:**

(a) In case of writing the instructions to the RAM before jumping to the RAM to execute instructions from the RAM, take the following procedure.

(1) Execute the store instruction to update a memory (ST.W, etc).

(2) Perform a dummy read of the above memory (LD.W, etc).

(3) Execute SYNCP.

(4) Execute SYNCI.

(5) Execute the subsequent instruction (branch instruction, etc).

(b) In case of updating control registers for memory protection and ECC functions before jumping to the memory to be controlled by the registers, take the following procedure.

(1) Execute the store instruction to update a control register (ST.W, etc).

(2) Perform a dummy read of the control register (LD.W, etc).

(3) Execute SYNCP.

(4) Execute SYNCI.

(5) Execute the subsequent instruction (branch instruction, etc).

**When switching the code flash memory area:**

In this case, see *Section 10, Usage Notes, (7) Updating the BFASELR register* in the *RH850/F1K Flash Memory User's Manual: Hardware Interface*.

### 3.3.2 Ensure Coherency after Rewriting the Code Flash

The CPU1 is equipped with the buffer for the code flash area as described in **Section 3.2.2, Buffers for Code Flash**.

Therefore, clear the buffer to ensure coherency after rewriting the code flash by self-programming.

### 3.3.3 Access to Registers by Using Bit-Manipulation Instructions

Writing bit-manipulation instructions consists of read-modify-write processing in 8-bit units. Thus, access by a bit-manipulation instruction is only possible for registers for which reading and writing in 8-bit units is possible. If a register includes multiple flag bits, the read-modify-write operation may lead to the clearing of flags that were not actually targets for clearing.

### 3.3.4 Caution of Prefetching

There is a possibility that the reading of the memory occurs by pre-fetch from the area where instruction codes do not exist. Secure more than 40-byte initialized area after the area where instruction codes are stored.

# Section 4 Address Space

## 4.1 Address Space

**Table 4.1**, **Table 4.2**, **Table 4.3** and **Table 4.4** show the address space of the RH850/F1K.

**CAUTION**

**Do not access an address with which no register is mapped in the on-chip I/O register space. In addition, do not access any access prohibited area specified in Table 4.1, Table 4.2, Table 4.3 and Table 4.4. If such an address is accessed, operation is not guaranteed.**

**NOTE**

CPU1 area: Address area accessible from CPU and DMA

Self area: Address area accessible only from CPU to refer the CPU's self resource

**Table 4.1 Address Space (2-MB 100, 144, 176 Pin Product)**

| Address | Address Space Type | Size |
|---|---|---|
| 0000 $0000_H$ to 001F $FFFF_H$ | Code flash | 2 MB |
| 0020 $0000_H$ to 00FF $FFFF_H$ | Access prohibited area | |
| 0100 $0000_H$ to 0100 $7FFF_H$ | Code flash (extended user area) | 32 KB |
| 0100 $8000_H$ to FEBC $FFFF_H$ | Access prohibited area | |
| FEBD $0000_H$ to FEBE $FFFF_H$ | Local RAM (CPU1 area) | 128 KB |
| FEBF $0000_H$ to FEBF $FFFF_H$ | Retention RAM (CPU1 area) | 64 KB |
| FEC0 $0000_H$ to FEDC $FFFF_H$ | Access prohibited area | |
| FEDD $0000_H$ to FEDE $FFFF_H$ | Local RAM (self area) | 128 KB |
| FEDF $0000_H$ to FEDF $FFFF_H$ | Retention RAM (self area) | 64 KB |
| FEE0 $0000_H$ to FF1F $FFFF_H$ | Access prohibited area | |
| FF20 $0000_H$ to FF20 $FFFF_H$ | Data flash | 64 KB |
| FF21 $0000_H$ to FF9F $FFFF_H$ | Access prohibited area | |
| FFA0 $0000_H$ to FFFD $FFFF_H$ | On-chip peripheral I/O area | 6 MB - 128 KB |
| FFFE $0000_H$ to FFFE $DFFF_H$ | Access prohibited area | |
| FFFE $E000_H$ to FFFE $FFFF_H$ | On-chip peripheral I/O area (self area) | 8 KB |
| FFFF $0000_H$ to FFFF $4FFF_H$ | Access prohibited area | |
| FFFF $5000_H$ to FFFF $FFFF_H$ | On-chip peripheral I/O area | 44 KB |

Table 4.2 Address Space (1.5-MB 100, 144, 176 Pin Product)

| Address | Address Space Type | Size |
|---|---|---|
| 0000 0000$_H$ to 0017 FFFF$_H$ | Code flash | 1.5 MB |
| 0018 0000$_H$ to 00FF FFFF$_H$ | Access prohibited area | |
| 0100 0000$_H$ to 0100 7FFF$_H$ | Code flash (extended user area) | 32 KB |
| 0100 8000$_H$ to FEBD 7FFF$_H$ | Access prohibited area | |
| FEBD 8000$_H$ to FEBE FFFF$_H$ | Local RAM (CPU1 area) | 96 KB |
| FEBF 0000$_H$ to FEBF FFFF$_H$ | Retention RAM (CPU1 area) | 64 KB |
| FEC0 0000$_H$ to FEDD 7FFF$_H$ | Access prohibited area | |
| FEDD 8000$_H$ to FEDE FFFF$_H$ | Local RAM (self area) | 96 KB |
| FEDF 0000$_H$ to FEDF FFFF$_H$ | Retention RAM (self area) | 64 KB |
| FEE0 0000$_H$ to FF1F FFFF$_H$ | Access prohibited area | |
| FF20 0000$_H$ to FF20 FFFF$_H$ | Data flash | 64 KB |
| FF21 0000$_H$ to FF9F FFFF$_H$ | Access prohibited area | |
| FFA0 0000$_H$ to FFFD FFFF$_H$ | On-chip peripheral I/O area | 6 MB - 128 KB |
| FFFE 0000$_H$ to FFFE DFFF$_H$ | Access prohibited area | |
| FFFE E000$_H$ to FFFE FFFF$_H$ | On-chip peripheral I/O area (self area) | 8 KB |
| FFFF 0000$_H$ to FFFF 4FFF$_H$ | Access prohibited area | |
| FFFF 5000$_H$ to FFFF FFFF$_H$ | On-chip peripheral I/O area | 44 KB |

Table 4.3 Address Space (1-MB 100, 144, 176 Pin Product)

| Address | Address Space Type | Size |
|---|---|---|
| 0000 0000$_H$ to 000F FFFF$_H$ | Code flash | 1 MB |
| 0010 0000$_H$ to 00FF FFFF$_H$ | Access prohibited area | |
| 0100 0000$_H$ to 0100 7FFF$_H$ | Code flash (extended user area) | 32 KB |
| 0100 8000$_H$ to FEBD FFFF$_H$ | Access prohibited area | |
| FEBE 0000$_H$ to FEBE FFFF$_H$ | Local RAM (CPU1 area) | 64 KB |
| FEBF 0000$_H$ to FEBF FFFF$_H$ | Retention RAM (CPU1 area) | 64 KB |
| FEC0 0000$_H$ to FEDD FFFF$_H$ | Access prohibited area | |
| FEDE 0000$_H$ to FEDE FFFF$_H$ | Local RAM (self area) | 64 KB |
| FEDF 0000$_H$ to FEDF FFFF$_H$ | Retention RAM (self area) | 64 KB |
| FEE0 0000$_H$ to FF1F FFFF$_H$ | Access prohibited area | |
| FF20 0000$_H$ to FF20 FFFF$_H$ | Data flash | 64 KB |
| FF21 0000$_H$ to FF9F FFFF$_H$ | Access prohibited area | |
| FFA0 0000$_H$ to FFFD FFFF$_H$ | On-chip peripheral I/O area | 6 MB - 128 KB |
| FFFE 0000$_H$ to FFFE DFFF$_H$ | Access prohibited area | |
| FFFE E000$_H$ to FFFE FFFF$_H$ | On-chip peripheral I/O area (self area) | 8 KB |
| FFFF 0000$_H$ to FFFF 4FFF$_H$ | Access prohibited area | |
| FFFF 5000$_H$ to FFFF FFFF$_H$ | On-chip peripheral I/O area | 44 KB |

Table 4.4 Address Space (768-KB 100, 144 Pin Product)

| Address | Address Space Type | Size |
|---|---|---|
| 0000 $0000_H$ to 000B $FFFF_H$ | Code flash | 768 KB |
| 000C $0000_H$ to 00FF $FFFF_H$ | Access prohibited area | |
| 0100 $0000_H$ to 0100 $7FFF_H$ | Code flash (extended user area) | 32 KB |
| 0100 $8000_H$ to FEBE $7FFF_H$ | Access prohibited area | |
| FEBE $8000_H$ to FEBE $FFFF_H$ | Local RAM (CPU1 area) | 32 KB |
| FEBF $0000_H$ to FEBF $FFFF_H$ | Retention RAM (CPU1 area) | 64 KB |
| FEC0 $0000_H$ to FEDE $7FFF_H$ | Access prohibited area | |
| FEDE $8000_H$ to FEDE $FFFF_H$ | Local RAM (self area) | 32 KB |
| FEDF $0000_H$ to FEDF $FFFF_H$ | Retention RAM (self area) | 64 KB |
| FEE0 $0000_H$ to FF1F $FFFF_H$ | Access prohibited area | |
| FF20 $0000_H$ to FF20 $FFFF_H$ | Data flash | 64 KB |
| FF21 $0000_H$ to FF9F $FFFF_H$ | Access prohibited area | |
| FFA0 $0000_H$ to FFFD $FFFF_H$ | On-chip peripheral I/O area | 6 MB - 128 KB |
| FFFE $0000_H$ to FFFE $DFFF_H$ | Access prohibited area | |
| FFFE $E000_H$ to FFFE $FFFF_H$ | On-chip peripheral I/O area (self area) | 8 KB |
| FFFF $0000_H$ to FFFF $4FFF_H$ | Access prohibited area | |
| FFFF $5000_H$ to FFFF $FFFF_H$ | On-chip peripheral I/O area | 44 KB |

## 4.2 Address Space Viewed from Each Bus Master

**Table 4.5** shows address spaces viewed from each bus master.

### 4.2.1 Space in which Instructions can be Fetched

Instructions of the CPU can be fetched from the Code flash, Local RAM, and Retention RAM.

### 4.2.2 Data Space Accessible by CPU

See **Table 4.5** for the spaces accessible from the CPU.

### 4.2.3 Data Space Accessible by DMA

See **Table 4.5** for the spaces accessible from the DMA.

**Table 4.5 Address Space Viewed from Each Bus Master (2-MB Product)**

| Address | Access from CPU | Access from DMA |
|---|---|---|
| 0000 0000$_H$ to 001F FFFF$_H$ | Code flash | Code flash |
| 0020 0000$_H$ to 00FF FFFF$_H$ | Access prohibited area | Access prohibited area |
| 0100 0000$_H$ to 0100 7FFF$_H$ | Code flash (extended user area) | Code flash (extended user area) |
| 0100 8000$_H$ to FEBC FFFF$_H$ | Access prohibited area | Access prohibited area |
| FEBD 0000$_H$ to FEBE FFFF$_H$ | Local RAM (CPU1 area) | Local RAM (CPU1 area) |
| FEBF 0000$_H$ to FEBF FFFF$_H$ | Retention RAM (CPU1 area) | Retention RAM (CPU1 area) |
| FEC0 0000$_H$ to FEDC FFFF$_H$ | Access prohibited area | Access prohibited area |
| FEDD 0000$_H$ to FEDE FFFF$_H$ | Local RAM (self area) | Access prohibited area |
| FEDF 0000$_H$ to FEDF FFFF$_H$ | Retention RAM (self area) | Access prohibited area |
| FEE0 0000$_H$ to FF1F FFFF$_H$ | Access prohibited area | Access prohibited area |
| FF20 0000$_H$ to FF20 FFFF$_H$ | Data flash | Data flash |
| FF21 0000$_H$ to FF9F FFFF$_H$ | Access prohibited area | Access prohibited area |
| FFA0 0000$_H$ to FFFD FFFF$_H$ | On-chip peripheral I/O area | On-chip peripheral I/O area |
| FFFE 0000$_H$ to FFFE DFFF$_H$ | Access prohibited area | Access prohibited area |
| FFFE E000$_H$ to FFFE FFFF$_H$ | On-chip peripheral I/O area (self area) | Access prohibited area |
| FFFF 0000$_H$ to FFFF 4FFF$_H$ | Access prohibited area | Access prohibited area |
| FFFF 5000$_H$ to FFFF FFFF$_H$ | On-chip peripheral I/O area | On-chip peripheral I/O area |

**Note:** The following color coding is used in the map above.

| |
|---|
| Fetch and data access available |
| Data access available |
| Access prohibited |

## 4.3 Peripheral I/O Address Map

**Table 4.6** shows peripheral I/O address map.

Table 4.6 Peripheral I/O Address Map (1/5)

| Address | Peripheral Group | Peripheral I/O |
|---|---|---|
| FF00 0000$_H$ to FF1F FFFF$_H$ | — | Access prohibited area |
| FF20 0000$_H$ to FF20 FFFF$_H$ | 2 | Data flash |
| FF21 0000$_H$ to FF9F FFFF$_H$ | — | Access prohibited area |
| FFA0 0000$_H$ to FFA0 001F$_H$ | 1 | FLMD (FLMDCNT) |
| FFA0 0020$_H$ to FFA0 0FFF$_H$ | — | Access prohibited area |
| FFA0 1000$_H$ to FFA0 103F$_H$ | — | Access prohibited area |
| FFA0 1040$_H$ to FFA0 1FFF$_H$ | — | Access prohibited area |
| FFA0 2000$_H$ to FFA0 201F$_H$ | — | Access prohibited area |
| FFA0 2020$_H$ to FFA0 FFFF$_H$ | — | Access prohibited area |
| FFA1 0000$_H$ to FFA1 1FFF$_H$ | 1 | Flash controller |
| FFA1 2000$_H$ to FFA1 FFFF$_H$ | — | Access prohibited area |
| FFA2 0000$_H$ to FFA2 FFFF$_H$ | 1 | FACI command-issuing area |
| FFA3 0000$_H$ to FFBF FFFF$_H$ | — | Access prohibited area |
| FFC0 0000$_H$ to FFC0 000F$_H$ | 1 | FENMI |
| FFC0 0010$_H$ to FFC0 00FF$_H$ | — | Access prohibited area |
| FFC0 0100$_H$ to FFC0 010F$_H$ | 1 | FEINT |
| FFC0 0110$_H$ to FFC0 0FFF$_H$ | — | Access prohibited area |
| FFC0 1000$_H$ to FFC0 1007$_H$ | 1 | SELB_INTC |
| FFC0 1008$_H$ to FFC0 FFFF$_H$ | — | Access prohibited area |
| FFC1 0000$_H$ to FFC1 4CCF$_H$ | 1 | PORT |
| FFC1 4CD0$_H$ to FFC1 FFFF$_H$ | — | Access prohibited area |
| FFC2 0000$_H$ to FFC2 04CF$_H$ | 1 | PORT (JTAG) |
| FFC2 04D0$_H$ to FFC2 FFFF$_H$ | — | Access prohibited area |
| FFC3 0000$_H$ to FFC3 00CF$_H$ | 1 | PORT (DNF) |
| FFC3 00D0$_H$ to FFC3 3FFF$_H$ | — | Access prohibited area |
| FFC3 4000$_H$ to FFC3 409F$_H$ | 1 | FCLA0 |
| FFC3 40A0$_H$ to FFC3 FFFF$_H$ | — | Access prohibited area |
| FFC4 0000$_H$ to FFC4 004F$_H$ | 1 | P-Bus guard (PBG10) |
| FFC4 0050$_H$ to FFC4 00FF$_H$ | — | Access prohibited area |
| FFC4 0100$_H$ to FFC4 014F$_H$ | 1 | P-Bus guard (PBG11) |
| FFC4 0150$_H$ to FFC4 03FF$_H$ | — | Access prohibited area |
| FFC4 0400$_H$ to FFC4 044F$_H$ | 1 | P-Bus guard (PBG12) |
| FFC4 0450$_H$ to FFC4 04FF$_H$ | — | Access prohibited area |
| FFC4 0500$_H$ to FFC4 054F$_H$ | 1 | P-Bus guard (PBG13) |
| FFC4 0550$_H$ to FFC4 BFFF$_H$ | — | Access prohibited area |
| FFC4 C000$_H$ to FFC4 C00F$_H$ | 2 | P-Bus guard (PBGC0) |
| FFC4 C010$_H$ to FFC4 C0FF$_H$ | — | Access prohibited area |
| FFC4 C100$_H$ to FFC4 C13F$_H$ | 2 | P-Bus guard (PBGC1) |
| FFC4 C140$_H$ to FFC4 C7FF$_H$ | — | Access prohibited area |
| FFC4 C800$_H$ to FFC4 C80F$_H$ | 2 | ERRSLV (PBGC0) |
| FFC4 C810$_H$ to FFC4 C8FF$_H$ | — | Access prohibited area |

Table 4.6 Peripheral I/O Address Map (2/5)

| Address | Peripheral Group | Peripheral I/O |
|---|---|---|
| FFC4 C900$_H$ to FFC4 C90F$_H$ | 2 | ERRSLV (PBGC1) |
| FFC4 C910$_H$ to FFC5 97FF$_H$ | — | Access prohibited area |
| FFC5 9800$_H$ to FFC5 981F$_H$ | 2 | DCIB (EEPRDCYCL) |
| FFC5 9820$_H$ to FFC5 AFFF$_H$ | — | Access prohibited area |
| FFC5 B000$_H$ to FFC5 B00F$_H$ | 2 | FBUF CTRL |
| FFC5 B010$_H$ to FFC6 21FF$_H$ | — | Access prohibited area |
| FFC6 2200$_H$ to FFC6 23FF$_H$ | 2 | Code flash ECC (VCI) |
| FFC6 2400$_H$ to FFC6 25FF$_H$ | 2 | Code flash ECC (CPU1) |
| FFC6 2600$_H$ to FFC6 29FF$_H$ | — | Access prohibited area |
| FFC6 2A00$_H$ to FFC6 2A3F$_H$ | 2 | Data flash ECC |
| FFC6 2A40$_H$ to FFC6 4FFF$_H$ | — | Access prohibited area |
| FFC6 5000$_H$ to FFC6 501F$_H$ | 2 | Local RAM ECC TEST |
| FFC6 5020$_H$ to FFC6 53FF$_H$ | — | Access prohibited area |
| FFC6 5400$_H$ to FFC6 55FF$_H$ | 2 | Local RAM ECC |
| FFC6 5600$_H$ to FFC7 00FF$_H$ | — | Access prohibited area |
| FFC7 0100$_H$ to FFC7 0413$_H$ | 3 | ECC CSIH0-3 |
| FFC7 0414$_H$ to FFC7 12FF$_H$ | — | Access prohibited area |
| FFC7 1300$_H$ to FFC7 1413$_H$ | 3 | ECC CAN |
| FFC7 1414$_H$ to FFC7 7FFF$_H$ | — | Access prohibited area |
| FFC7 8000$_H$ to FFC7 8007$_H$ | 1 | SELB READ TEST |
| FFC7 8008$_H$ to FFC9 FFFF$_H$ | — | Access prohibited area |
| FFCA 0000$_H$ to FFCA 007F$_H$ | 1 | RIIC0 |
| FFCA 0080$_H$ to FFCC FFFF$_H$ | — | Access prohibited area |
| FFCD 0000$_H$ to FFCD 01FF$_H$ | 1 | SCDS (PRDNAME/CHIPID) |
| FFCD 0200$_H$ to FFCD FFFF$_H$ | — | Access prohibited area |
| FFCE 0000$_H$ to FFCE 017F$_H$ | 1 | RLIN2 |
| FFCE 0180$_H$ to FFCE 1FFF$_H$ | — | Access prohibited area |
| FFCE 2000$_H$ to FFCE 217F$_H$ | 1 | RLIN3 |
| FFCE 2180$_H$ to FFCF FFFF$_H$ | — | Access prohibited area |
| FFD0 0000$_H$ to FFD0 7FFF$_H$ | 3 | RSCAN0 |
| FFD0 8000$_H$ to FFD0 FFFF$_H$ | 3 | RSCAN1 |
| FFD1 0000$_H$ to FFD6 CFFF$_H$ | — | Access prohibited area |
| FFD6 D000$_H$ to FFD6 D7FF$_H$ | 3 | ADCA1 |
| FFD6 D800$_H$ to FFD6 FFFF$_H$ | — | Access prohibited area |
| FFD7 0000$_H$ to FFD7 003F$_H$ | 3 | OSTM0 |
| FFD7 0040$_H$ to FFD7 00FF$_H$ | — | Access prohibited area |
| FFD7 0100$_H$ to FFD7 013F$_H$ | 3 | OSTM1 |
| FFD7 0140$_H$ to FFD7 01FF$_H$ | — | Access prohibited area |
| FFD7 0200$_H$ to FFD7 023F$_H$ | 3 | OSTM2 |
| FFD7 0240$_H$ to FFD7 02FF$_H$ | — | Access prohibited area |
| FFD7 0300$_H$ to FFD7 033F$_H$ | 3 | OSTM3 |
| FFD7 0340$_H$ to FFD7 03FF$_H$ | — | Access prohibited area |
| FFD7 0400$_H$ to FFD7 043F$_H$ | 3 | OSTM4 |
| FFD7 0440$_H$ to FFD7 FFFF$_H$ | — | Access prohibited area |

Table 4.6 Peripheral I/O Address Map (3/5)

| Address | Peripheral Group | Peripheral I/O |
|---|---|---|
| FFD8 0000$_H$ to FFD8 107F$_H$ | 3 | CSIH0 |
| FFD8 1080$_H$ to FFD8 1FFF$_H$ | — | Access prohibited area |
| FFD8 2000$_H$ to FFD8 307F$_H$ | 3 | CSIH1 |
| FFD8 3080$_H$ to FFD8 3FFF$_H$ | — | Access prohibited area |
| FFD8 4000$_H$ to FFD8 507F$_H$ | 3 | CSIH2 |
| FFD8 5080$_H$ to FFD8 5FFF$_H$ | — | Access prohibited area |
| FFD8 6000$_H$ to FFD8 707F$_H$ | 3 | CSIH3 |
| FFD8 7080$_H$ to FFD8 7FFF$_H$ | — | Access prohibited area |
| FFD8 8000$_H$ to FFD8 901F$_H$ | 3 | CSIG0 |
| FFD8 9020$_H$ to FFD8 9FFF$_H$ | — | Access prohibited area |
| FFD8 A000$_H$ to FFD8 B01F$_H$ | 3 | CSIG1 |
| FFD8 B020$_H$ to FFDC FFFF$_H$ | — | Access prohibited area |
| FFDD 0000$_H$ to FFDD 00FF$_H$ | 2 | PIC0 |
| FFDD 0100$_H$ to FFDD CFFF$_H$ | — | Access prohibited area |
| FFDD D000$_H$ to FFDD D04F$_H$ | 2 | P-Bus guard (PBG20) |
| FFDD D050$_H$ to FFDD D0FF$_H$ | — | Access prohibited area |
| FFDD D100$_H$ to FFDD D14F$_H$ | 2 | P-Bus guard (PBG21) |
| FFDD D150$_H$ to FFE1 FFFF$_H$ | — | Access prohibited area |
| FFE2 0000$_H$ to FFE2 03FF$_H$ | 2 | TAUD0 |
| FFE2 0400$_H$ to FFE2 3FFF$_H$ | — | Access prohibited area |
| FFE2 4000$_H$ to FFE2 4007$_H$ | 2 | SELB_TAUD0 |
| FFE2 4008$_H$ to FFE2 FFFF$_H$ | — | Access prohibited area |
| FFE3 0000$_H$ to FFE3 03FF$_H$ | 2 | TAUB0 |
| FFE3 0400$_H$ to FFE3 0FFF$_H$ | — | Access prohibited area |
| FFE3 1000$_H$ to FFE3 13FF$_H$ | 2 | TAUB1 |
| FFE3 1400$_H$ to FFE4 FFFF$_H$ | — | Access prohibited area |
| FFE5 0000$_H$ to FFE5 00FF$_H$ | 2 | TAUJ0 |
| FFE5 0100$_H$ to FFE5 0FFF$_H$ | — | Access prohibited area |
| FFE5 1000$_H$ to FFE5 10FF$_H$ | 2 | TAUJ1 |
| FFE5 1100$_H$ to FFE5 3FFF$_H$ | — | Access prohibited area |
| FFE5 4000$_H$ to FFE5 4007$_H$ | 2 | SELB_TAUJ0 |
| FFE5 4008$_H$ to FFE6 FFFF$_H$ | — | Access prohibited area |
| FFE7 0000$_H$ to FFE7 00FF$_H$ | 2 | PWSA0 |
| FFE7 0100$_H$ to FFE7 0FFF$_H$ | — | Access prohibited area |
| FFE7 1000$_H$ to FFE7 21FF$_H$ | 2 | PWGA |
| FFE7 2200$_H$ to FFE7 27FF$_H$ | — | Access prohibited area |
| FFE7 2800$_H$ to FFE7 281F$_H$ | 2 | PWBA0 |
| FFE7 2820$_H$ to FFE7 2FFF$_H$ | — | Access prohibited area |
| FFE7 3000$_H$ to FFE7 300F$_H$ | 2 | SLPWGA |
| FFE7 3010$_H$ to FFE7 7FFF$_H$ | — | Access prohibited area |
| FFE7 8000$_H$ to FFE7 807F$_H$ | 2 | RTCA0 |
| FFE7 8080$_H$ to FFE7 FFFF$_H$ | — | Access prohibited area |
| FFE8 0000$_H$ to FFE8 007F$_H$ | 2 | ENCA0 |
| FFE8 0080$_H$ to FFE8 FFFF$_H$ | — | Access prohibited area |

Table 4.6 Peripheral I/O Address Map (4/5)

| Address | Peripheral Group | Peripheral I/O |
|---|---|---|
| FFE9 0000$_H$ to FFE9 003F$_H$ | 2 | TAPA0 |
| FFE9 0040$_H$ to FFEC FFFF$_H$ | — | Access prohibited area |
| FFED 0000$_H$ to FFED 000F$_H$ | 2 | WDTA0 |
| FFED 0010$_H$ to FFED 0FFF$_H$ | — | Access prohibited area |
| FFED 1000$_H$ to FFED 100F$_H$ | 2 | WDTA1 |
| FFED 1010$_H$ to FFF1 FFFF$_H$ | — | Access prohibited area |
| FFF2 0000$_H$ to FFF2 07FF$_H$ | 1 | ADCA0 |
| FFF2 0800$_H$ to FFF6 FFFF$_H$ | — | Access prohibited area |
| FFF7 0000$_H$ to FFF7 003F$_H$ | 1 | DCRA0 |
| FFF7 0040$_H$ to FFF7 0FFF$_H$ | — | Access prohibited area |
| FFF7 1000$_H$ to FFF7 103F$_H$ | 1 | DCRA1 |
| FFF7 1040$_H$ to FFF7 1FFF$_H$ | — | Access prohibited area |
| FFF7 2000$_H$ to FFF7 203F$_H$ | 1 | DCRA2 |
| FFF7 2040$_H$ to FFF7 2FFF$_H$ | — | Access prohibited area |
| FFF7 3000$_H$ to FFF7 303F$_H$ | 1 | DCRA3 |
| FFF7 3040$_H$ to FFF7 7FFF$_H$ | — | Access prohibited area |
| FFF7 8000$_H$ to FFF7 8003$_H$ | 1 | KR0 |
| FFF7 8004$_H$ to FFF7 FFFF$_H$ | — | Access prohibited area |
| FFF8 0000$_H$ to FFF8 000F$_H$ | 1 | Write protected register (WPROTR) |
| FFF8 0010$_H$ to FFF8 00FF$_H$ | — | Access prohibited area |
| FFF8 0100$_H$ to FFF8 011F$_H$ | 1 | STBC0 |
| FFF8 0120$_H$ to FFF8 03FF$_H$ | — | Access prohibited area |
| FFF8 0400$_H$ to FFF8 040F$_H$ | 1 | STBC_WUF0 |
| FFF8 0410$_H$ to FFF8 051F$_H$ | — | Access prohibited area |
| FFF8 0520$_H$ to FFF8 052F$_H$ | 1 | STBC_WUF20 |
| FFF8 0530$_H$ to FFF8 075F$_H$ | — | Access prohibited area |
| FFF8 0760$_H$ to FFF8 0AFF$_H$ | 1 | Reset controller/Supply voltage monitor (LVI, VLVI) |
| FFF8 0B00$_H$ to FFF8 0FFF$_H$ | 1 | STBC_IOHOLD |
| FFF8 1000$_H$ to FFF8 2FFF$_H$ | 1 | Clock controller (CLKCTL) |
| FFF8 3000$_H$ to FFF8 307F$_H$ | 1 | LPS0 |
| FFF8 3080$_H$ to FFF8 30FF$_H$ | — | Access prohibited area |
| FFF8 3100$_H$ to FFF8 3207$_H$ | 1 | Supply voltage monitor (CVM) |
| FFF8 3208$_H$ to FFF8 35FF$_H$ | — | Access prohibited area |
| FFF8 3600$_H$ to FFF8 3603$_H$ | 1 | Reset controller (CYCRBASE) |
| FFF8 3604$_H$ to FFF8 7FFF$_H$ | — | Access prohibited area |
| FFF8 8000$_H$ to FFF8 800F$_H$ | 1 | Write protected register (WPROTR) |
| FFF8 8010$_H$ to FFF8 810F$_H$ | — | Access prohibited area |
| FFF8 8110$_H$ to FFF8 811F$_H$ | 1 | STBC_WUFISO |
| FFF8 8120$_H$ to FFF8 BFFF$_H$ | 1 | Clock controller (CLKCTL) |
| FFF8 C000$_H$ to FFF8 CFFF$_H$ | 1 | CLMA0 |
| FFF8 D000$_H$ to FFF8 DFFF$_H$ | 1 | CLMA1 |
| FFF8 E000$_H$ to FFF8 EFFF$_H$ | 1 | CLMA2 |
| FFF8 F000$_H$ to FFF8 FFFF$_H$ | — | Access prohibited area |

Table 4.6 Peripheral I/O Address Map (5/5)

| Address | Peripheral Group | Peripheral I/O |
|---|---|---|
| FFF9 0000$_H$ to FFF9 004F$_H$ | 1 | P-Bus guard (PBG50) |
| FFF9 0050$_H$ to FFF9 3FFF$_H$ | — | Access prohibited area |
| FFF9 4000$_H$ to FFF9 404F$_H$ | 3 | P-Bus guard (PBG30) |
| FFF9 4050$_H$ to FFF9 40FF$_H$ | — | Access prohibited area |
| FFF9 4100$_H$ to FFF9 414F$_H$ | 3 | P-Bus guard (PBG31) |
| FFF9 4150$_H$ to FFF9 41FF$_H$ | — | Access prohibited area |
| FFF9 4200$_H$ to FFF9 424F$_H$ | 3 | P-Bus guard (PBG32) |
| FFF9 4250$_H$ to FFFE DFFF$_H$ | — | Access prohibited area |
| FFFE E000$_H$ to FFFE E0FF$_H$ | CPU local peripheral | Access prohibited area |
| FFFE E100$_H$ to FFFE E5FF$_H$ | | Access prohibited area |
| FFFE E600$_H$ to FFFE E6FF$_H$ | | PEG |
| FFFE E700$_H$ to FFFE E97F$_H$ | | Access prohibited area |
| FFFE E980$_H$ to FFFE E9FF$_H$ | | SEG |
| FFFE EA00$_H$ to FFFE EBFF$_H$ | | INTC1 |
| FFFE EC00$_H$ to FFFF 4FFF$_H$ | | Access prohibited area |
| FFFF 5000$_H$ to FFFF 7FFF$_H$ | — | Access prohibited area |
| FFFF 8000$_H$ to FFFF AFFF$_H$ | 2 | DMAC |
| FFFF B000$_H$ to FFFF BFFF$_H$ | 2 | INTC2 |
| FFFF C000$_H$ to FFFF FFFF$_H$ | — | Access prohibited area |

# Section 5 Write-Protected Registers

This section contains a generic description of the write-protected registers.

The first part in this section describes the features specific to the write-protected registers, and the ensuing sections describe the various registers.

## 5.1 Overview

### 5.1.1 Functional Overview

The RH850/F1K products require a special procedure using write-protected registers to set important registers that affect the system, such as clock, reset, and port-related registers. The settings of protected registers are protected against illegal writing by programs by requiring a special procedure. For details about the protected registers, see **Section 5.1.5, Write-Protection Target Registers**. Write-protected registers are managed in units of protected registers called register protection clusters.

### 5.1.2 Writing Procedure to Write-Protected Registers

Write access to a write-protected register is enabled by using the following protection unlock sequence:

1. Write the fixed value 0000 $00A5_H$ to the protection command register.
2. Write the desired value to the protected register.
3. Write the bit-wise inversion of the desired value to the protected register.
4. Write the desired value to the protected register.
5. Verify that the desired value has been written to the protected register.
   Verify successful write of the desired value to the protected register by verifying that the error monitor bit in the protection status register is “0”.
   In case the write was not successful, indicated by the error monitor bit set to “1”, the entire sequence has to be restarted at step 1.

If another register (second register) is accessed between step 1 and step 4 of the above sequence for writing to a write-protected register (first register), the protection mechanism operates as follows:

- If the second register belongs to the same cluster, the write to the protected register fails (the error monitor bit is set to 1). The entire sequence has to be restarted at step 1.
- If the second register does not belong to the same cluster, the protection unlock sequence is not disrupted and the write to the first register completes successfully.

### 5.1.3 Interrupt during Write Protection Unlock

If an interrupt occurs during the protection unlock sequence, the protection mechanism operates as follows:

(1) If an interrupt request is accepted during the protection unlock sequence and write access to a register of the same cluster is performed

The protection unlock sequence is disrupted, so the write operation to the protected register cannot be completed upon returning from the interrupt service routine. **Figure 5.1** shows an execution example.

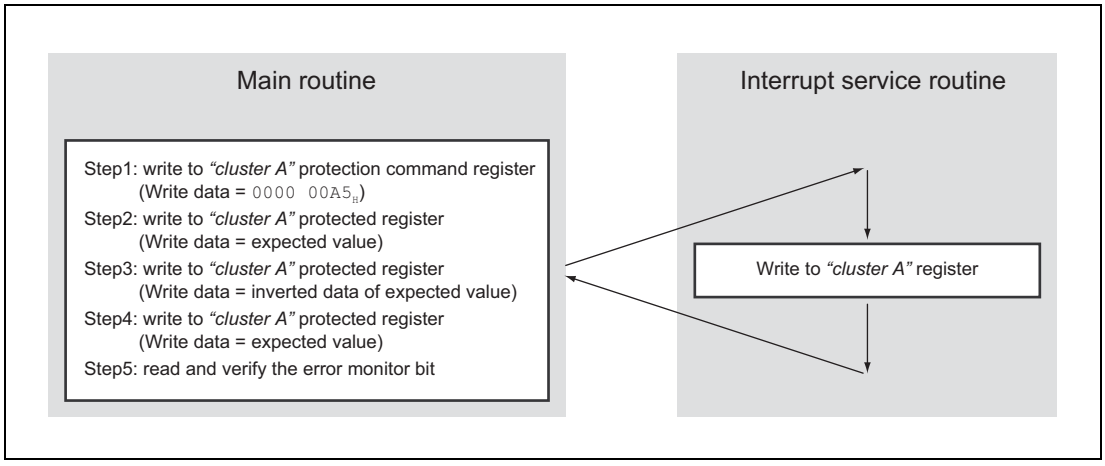



**Figure 5.1 Example of Interruption of Register Protection Unlock Sequence**

(2) If an interrupt request is accepted during the protection unlock sequence and write access to a register of a different cluster is performed

The protection unlock sequence is not disrupted, so the write operation to the protected register is completed upon returning from the interrupt service routine. **Figure 5.2** shows an execution example.

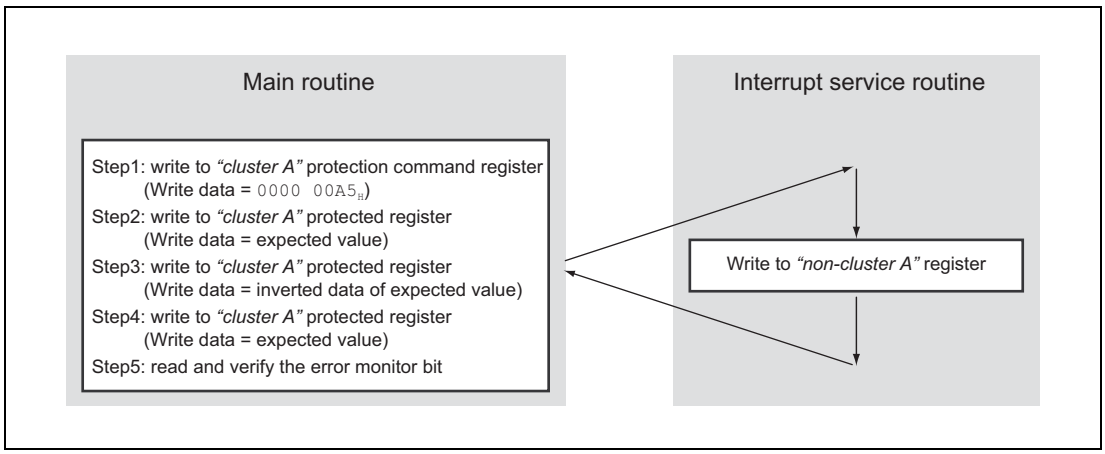



**Figure 5.2 Example of Successful Protection Unlock Sequence**

For more information on registers of RH850/F1K register protection clusters, see **Section 5.1.5, Write-Protection Target Registers**.

### 5.1.4 Emulation Break during Write Protection Unlock Sequence

If an emulation break occurs during the protection unlock sequence, e.g. because of a breakpoint hit, the register protection is suspended until normal operation is resumed.
Even if any register of the same cluster is accessed during the break, the protection unlock sequence is not disrupted and the error monitor bit is not set to 1.

### 5.1.5 Write-Protection Target Registers

The registers that are protected through the write-protection control registers are listed below.

**Table 5.1 Write-Protection Target Registers (1/2)**

| Protection Target | Protection Target Register | Protection Control Register | | Protection Cluster |
|---|---|---|---|---|
| | | Command Register | Status Register | |
| Clock controller | MOSCE | PROTCMD0 | PROTS0 | Control protection cluster 0 |
| | ROSCE | | | |
| | SOSCE | | | |
| | CKSC_AWDTAD_CTL | | | |
| | CKSC_ATAUJS_CTL | | | |
| | CKSC_ATAUJD_CTL | | | |
| | CKSC_ARTCAS_CTL | | | |
| | CKSC_ARTCAD_CTL | | | |
| | CKSC_AADCAS_CTL | | | |
| | CKSC_AADCAD_CTL | | | |
| | CKSC_AFOUTS_CTL | | | |
| | ROSCUT | | | |
| Stand-by function | STBC0PSC | | | |
| | STBC0STPT | | | |
| | IOHOLD | | | |
| Reset function | LVICNT | | | |
| | SWRESA | | | |
| | CYCRBASE | | | |
| Clock Controller | PLLE | PROTCMD1 | PROTS1 | Control protection cluster 1 |
| | CKSC_CPUCLKS_CTL | | | |
| | CKSC_CPUCLKD_CTL | | | |
| | CKSC_IPERI1S_CTL | | | |
| | CKSC_IPERI2S_CTL | | | |
| | CKSC_ILINS_CTL | | | |
| | CKSC_IADCAS_CTL | | | |
| | CKSC_IADCAD_CTL | | | |
| | CKSC_ILIND_CTL | | | |
| | CKSC_ICANS_CTL | | | |
| | CKSC_ICANOSCD_CTL | | | |
| | CKSC_ICSIS_CTL | | | |
| | CKSC_IIICS_CTL | | | |
| | CKSC_PPLLCLKS_CTL | | | |
| | CKSC_PLLIS_CTL | | | |

Table 5.1 Write-Protection Target Registers (2/2)

| Protection Target | Protection Target Register | Protection Control Register | | Protection Cluster |
|---|---|---|---|---|
| | | Command Register | Status Register | |
| Clock Monitors | CLMA0CTL0 | CLMA0PCMD | CLMA0PS | Clock Monitor control protection cluster 0 |
| | CLMA1CTL0 | CLMA1PCMD | CLMA1PS | Clock Monitor control protection cluster 1 |
| | CLMA2CTL0 | CLMA2PCMD | CLMA2PS | Clock Monitor control protection cluster 2 |
| | CLMATEST | PROTCMDCLMA | PROTSCLMA | Clock Monitor test protection cluster |
| Port*1 | JPODC0 | JPPCMD0 | JPPROTS0 | Port protection cluster 0 |
| | PODC0 | PPCMD0 | PPROTS0 | |
| | PODC1 | PPCMD1 | PPROTS1 | |
| | PODC2 | PPCMD2 | PPROTS2 | |
| | PODC8 | PPCMD8 | PPROTS8 | |
| | JPDSC0 | JPPCMD0 | JPPROTS0 | |
| | PDSC0 | PPCMD0 | PPROTS0 | |
| | PDSC1 | PPCMD1 | PPROTS1 | |
| | PDSC2 | PPCMD2 | PPROTS2 | |
| | PODC9 | PPCMD9 | PPROTS9 | Port protection cluster 1 |
| | PODC10 | PPCMD10 | PPROTS10 | |
| | PODC11 | PPCMD11 | PPROTS11 | |
| | PODC12 | PPCMD12 | PPROTS12 | |
| | PODC18 | PPCMD18 | PPROTS18 | |
| | PODC20 | PPCMD20 | PPROTS20 | |
| | PDSC10 | PPCMD10 | PPROTS10 | |
| | PDSC11 | PPCMD11 | PPROTS11 | |
| | PDSC12 | PPCMD12 | PPROTS12 | |
| | PDSC18 | PPCMD18 | PPROTS18 | |
| | PDSC20 | PPCMD20 | PPROTS20 | |
| Core Voltage Monitor | CVMF | PROTCMDCVM | PROTSCVM | Core Voltage Monitor protection cluster |
| | CVMDIAG | | | |
| Self-programming function | FLMDCNT | FLMDPCMD | FLMDPS | Self-programming protection cluster |

Note 1. Each port group has its own protection command register and status register. For details, see **(1) Port protection clusters** on the next page.

### (1) Port protection clusters

The following port control registers have a write protection function:

- Port open drain control registers (PODCn, JPODC0)
- Port drive strength control registers (PDSCn, JPDSC0)

The write protected port registers are divided into two port protection clusters as shown in the following table:

**Table 5.2 Port Protection Clusters**

| Port Protection Cluster | Port Group |
|---|---|
| 0 | JP0, P0, P1, P2, P8 |
| 1 | P9, P10, P11, P12, P18, P20 |

**NOTE**

Each port group n has its own port protection command register PPCMDn and port protection status register PPROTSn.

However, any port protection command registers of the same port protection cluster can be used in the protection unlock sequence. For instance, PPCMD1 can be used to unlock the protection of PODC2.

## 5.2 Registers

### 5.2.1 List of Registers

The following table lists the write-protection control registers.

**Table 5.3 List of Write-Protection Control Registers (1/2)**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| **Control protection clusters** | | | |
| WPROTR | Protection command register 0 | PROTCMD0 | FFF8 0000$_H$ |
| | Protection command register 1 | PROTCMD1 | FFF8 8000$_H$ |
| | Protection status register 0 | PROTS0 | FFF8 0004$_H$ |
| | Protection status register 1 | PROTS1 | FFF8 8004$_H$ |
| **Clock monitor control and test protection cluster** | | | |
| CLMAn | Protection command register 0 | CLMA0PCMD | FFF8 C010$_H$ |
| | Protection command register 1 | CLMA1PCMD | FFF8 D010$_H$ |
| | Protection command register 2 | CLMA2PCMD | FFF8 E010$_H$ |
| | Protection status register 0 | CLMA0PS | FFF8 C014$_H$ |
| | Protection status register 1 | CLMA1PS | FFF8 D014$_H$ |
| | Protection status register 2 | CLMA2PS | FFF8 E014$_H$ |
| CLMA | Protection command register | PROTCMDCLMA | FFF8 C200$_H$ |
| | Protection status register | PROTSCLMA | FFF8 C204$_H$ |
| **Port protection cluster 0** | | | |
| JTAG | Protection command registers | JPPCMD0 | FFC2 04C0$_H$ |
| PORT | | PPCMD0 | FFC1 4C00$_H$ |
| | | PPCMD1 | FFC1 4C04$_H$ |
| | | PPCMD2 | FFC1 4C08$_H$ |
| | | PPCMD8 | FFC1 4C20$_H$ |
| JTAG | Protection status registers | JPPROTS0 | FFC2 04B0$_H$ |
| PORT | | PPROTS0 | FFC1 4B00$_H$ |
| | | PPROTS1 | FFC1 4B04$_H$ |
| | | PPROTS2 | FFC1 4B08$_H$ |
| | | PPROTS8 | FFC1 4B20$_H$ |
| **Port protection cluster 1** | | | |
| PORT | Protection command registers | PPCMD9 | FFC1 4C24$_H$ |
| | | PPCMD10 | FFC1 4C28$_H$ |
| | | PPCMD11 | FFC1 4C2C$_H$ |
| | | PPCMD12 | FFC1 4C30$_H$ |
| | | PPCMD18 | FFC1 4C48$_H$ |
| | | PPCMD20 | FFC1 4C50$_H$ |

Table 5.3 List of Write-Protection Control Registers (2/2)

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| PORT | Protection status registers | PPROTS9 | FFC1 4B24$_H$ |
| | | PPROTS10 | FFC1 4B28$_H$ |
| | | PPROTS11 | FFC1 4B2C$_H$ |
| | | PPROTS12 | FFC1 4B30$_H$ |
| | | PPROTS18 | FFC1 4B48$_H$ |
| | | PPROTS20 | FFC1 4B50$_H$ |
| **Core Voltage Monitor protection cluster** | | | |
| SVM | Protection command register | PROTCMDCVM | FFF8 3200$_H$ |
| | Protection status register | PROTSCVM | FFF8 3204$_H$ |
| **Self-programming protection cluster** | | | |
| FLMD | Protection command register | FLMDPCMD | FFA0 0004$_H$ |
| | Protection error status register | FLMDPS | FFA0 0008$_H$ |

## 5.2.2 Details of Control Protection Cluster Registers

### 5.2.2.1 PROTCMDn — Protection Command Register

This register is used to initiate the write protection unlock sequence for write-protected registers.

#### Index n

An index “n” denotes the number of protection command registers. For details, see **Table 5.1, Write-Protection Target Registers**.

**Access:** This register is a write-only register that can be written in 32-bit units.

**Address:** See **Table 5.3, List of Write-Protection Control Registers**.

**Value after reset:** XXXX $XX00_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | PCMDn[7:0] | | | | | | | |
| Value after reset | — | — | — | — | — | — | — | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**Table 5.4 PROTCMDn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing to these bits, write 0. |
| 7 to 0 | PCMDn[7:0] | Protection command register bits to enable writing to protected registers of control protection cluster |

### 5.2.2.2 PROTSn — Protection Status Register

This register indicates the status of the protection unlock sequence performed by PROTCMDn.

#### Index n

An index “n” denotes the number of protection command registers. For details, see **Table 5.1, Write-Protection Target Registers**.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** See **Table 5.3, List of Write-Protection Control Registers**.

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PROTSnERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 5.5 PROTSn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | PROTSnERR | Write sequence protection error monitor<br>0: No protection error occurred<br>1: Protection error occurred |

## 5.2.3 Details of Clock Monitor Protection Cluster Registers

### 5.2.3.1 CLMAnPCMD — CLMAn Protection Command Register

This register is a protection command register for the CLMAnCTL0 register.

#### Index n

An index “n” denotes the number of protection command registers. For details, see **Table 5.1, Write-Protection Target Registers**.

**Access:** This register is a write-only register that can be written in 8-bit units.

**Address:** See **Table 5.3, List of Write-Protection Control Registers**.

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CLMAnREG[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W |

**Table 5.6 CLMAnPCMD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | CLMAnREG[7:0] | Protection command register bits to enable writing to the CLMAnCTL0 register |

### 5.2.3.2 CLMAnPS — CLMAn Protection Status Register

This register is used to verify whether the write-protected register (CLMAnCTL0) has been successfully written or not.

#### Index n

An index “n” denotes the number of protection command registers. For details, see **Table 5.1, Write-Protection Target Registers**.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** See **Table 5.3, List of Write-Protection Control Registers**.

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CLMAnPRERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 5.7 CLMAnPS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | CLMAnPRERR | Write protection sequence error monitor<br>0: No protection error occurred<br>1: Protection error occurred |

### 5.2.3.3 PROTCMDCLMA — Clock Monitor Test Protection Command Register

This register is a protection command register for the CLMATEST register.

**Access:** This register is a write-only register that can be written in 32-bit units.

**Address:** See **Table 5.3, List of Write-Protection Control Registers**.

**Value after reset:** XXXX XX00$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CLMATREG[7:0] | | | | | | | |
| Value after reset | — | — | — | — | — | — | — | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**Table 5.8 PROTCMDCLMA Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing to these bits, write “0”. |
| 7 to 0 | CLMATREG[7:0] | Protection command register bits to enable writing to the CLMATEST register |

### 5.2.3.4 PROTSCLMA — Clock Monitor Test Protection Status Register

This register is used to verify whether the write-protected register (CLMATEST) has been successfully written or not.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** See **Table 5.3, List of Write-Protection Control Registers**.

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CLMAT PRERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 5.9 PROTSCLMA Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | CLMATPRERR | Write protection sequence error monitor<br>0: No protection error occurred<br>1: Protection error occurred |

## 5.2.4 Details of Core Voltage Monitor Protection Cluster Registers

### 5.2.4.1 PROTCMDCVM — Core Voltage Monitor Protection Command Register

This register is a protection command register for the CVMF and CVMDIAG registers.

**Access:** This register is a write-only register that can be written in 32-bit units.

**Address:** See **Table 5.3, List of Write-Protection Control Registers**.

**Value after reset:** XXXX XX00$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CVMFREG[7:0] | | | | | | | |
| Value after reset | — | — | — | — | — | — | — | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**Table 5.10 PROTCMDCVM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing to these bits, write 0. |
| 7 to 0 | CVMFREG[7:0] | Protection command register bits to enable writing to the CVMF and CVMDIAG registers |

### 5.2.4.2 PROTSCVM — Core Voltage Monitor Protection Status Register

This register is used to verify whether the write-protected register (CVMF, CVMDIAG) has been successfully written or not.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** See **Table 5.3, List of Write-Protection Control Registers**.

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CVMFPRERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 5.11 PROTSCVM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | CVMFPRERR | Write protection sequence error monitor<br>0: No protection error occurred<br>1: Protection error occurred |

## 5.2.5 Details of Port Protection Cluster Registers

### 5.2.5.1 PPCMDn — Port Protection Command Register

PPCMDn is a protection command register for port group n.

#### Index n

An index “n” denotes the number of protection command registers. For details, see **Table 5.1, Write-Protection Target Registers**.

**Access:** This register is a write-only register that can be written in 32-bit units.

**Address:** See **Table 5.3, List of Write-Protection Control Registers**.

**Value after reset:** XXXX XX00$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | PPCMDn[7:0] | | | | | | | |
| Value after reset | — | — | — | — | — | — | — | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**NOTE**

The protection command register for port group JP0 is JPPCMD0. Its bits are JPPCMD0[7:0].

**Table 5.12 PPCMDn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing to these bits, write 0. |
| 7 to 0 | PPCMDn[7:0] | Protection command register bits that enable writing to port protection cluster registers |

### 5.2.5.2 PPROTSn — Port Protection Status Register

PPROTSn is a protection status register for write-protected registers of port group n. It indicates the status of the protection sequence operated by PPCMDn.

#### Index n

An index “n” denotes the number of protection command registers. For details, see **Table 5.1, Write-Protection Target Registers**.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** See **Table 5.3, List of Write-Protection Control Registers**.

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PPROTSnPRERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**NOTE**

The protection status register for port group JP0 is JPPROTS0. Its bit is JPPROTS0PRERR.

**Table 5.13 PPROTSn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | PPROTSn PRERR | Write protection sequence error monitor<br>0: No protection error occurred<br>1: Protection error occurred |

## 5.2.6 Details of Self-Programming Protection Cluster Registers

### 5.2.6.1 FLMDPCMD — FLMD Protection Command Register

FLMDPCMD is a protection command register for the FLMDCNT register.

**Access:** This register is a write-only register that can be written in 32-bit units.

**Address:** See **Table 5.3, List of Write-Protection Control Registers**.

**Value after reset:** XXXX XX00$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | FLMDPC[7:0] | | | | | | | |
| Value after reset | — | — | — | — | — | — | — | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**Table 5.14 FLMDPCMD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing to these bits, write 0. |
| 7 to 0 | FLMDPC[7:0] | Protection command register bits that enable writing to FLMDCNT register |

### 5.2.6.2 FLMDPS — FLMD Protection Error Status Register

This register is used to verify whether the write-protected register (FLMDCNT) has been successfully written or not.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** See **Table 5.3, List of Write-Protection Control Registers**.

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | FLMDPRERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 5.15 FLMDPS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | FLMDPRERR | Write protection sequence error monitor<br>0: No protection error occurred<br>1: Protection error occurred |

# Section 6 Operating Mode

This section describes the operating mode and mode selection of the RH850/F1K.

The RH850/F1K has the operating mode shown below.

- Normal operating mode

  This mode is for execution of the user program. The on-chip debug functions also use this mode. If FLMD0 is pulled up high during operation in this mode, writing to the code flash memory through self-programming is enabled.

- Serial programming mode

  The dedicated flash memory programmer enables erasing/writing to flash memory.

- Boundary scan mode

  This mode allows boundary scan tests compliant with IEEE Standard 1149.1.

- User boot mode

  This mode is the same as normal operating mode except that the reset vector base address is fixed to "$01000000_H$", and transition to standby modes is not supported.

When an external reset or power-on-clear reset is generated, the state of the FLMD0, FLMD1, MODE0, MODE1, and MODE2 pins are used to determine the operating mode after reset is released. The operating mode is fixed by the release of these reset factors. **Table 6.1** lists the relationship between the pin settings and the operating mode.

**Table 6.1 Selection of Operating Mode**

| Pins | | | | | Operating Mode |
|---|---|---|---|---|---|
| FLMD0 | FLMD1 (P10_8) | MODE0 (P10_1) | MODE1 (P10_2) | MODE2 (P10_6) | |
| 0 | x | x | x | x | Normal operating mode |
| 1 | 0 | x | x | x | Serial programming mode |
| 1 | 1 | 0 | 1 | x | Boundary scan mode |
| 1 | 1 | 1 | 1 | 1 | User boot mode |
| Other than the above | | | | | Setting prohibited |

**CAUTION**

**To change operating mode, restart from power-on clear reset. (Remove the power supply once and apply it again.) In the case of only by the external reset, some functions are not initialized after the mode transitions. For details of functions not initialized by the external reset, see Section 9.1.1, Reset Sources.**

# Section 7 Exception/Interrupts

## 7.1 Features

The act of branching from a currently running program to a different program in response to an event is called an exception. This microcontroller supports the following types of exceptions.

The details on exceptions are described in the *RH850G3KH User's Manual: Software*.

**Table 7.1 List of Exception Sources**

| Name | Symbol | Source | Priority | Saved to |
|---|---|---|---|---|
| Reset | RESET | Reset input | High | — |
| FE level non-maskable interrupt[*1] | FENMI | FENMI input | | FE |
| System error exception | SYSERR | SYSERR input | | FE |
| FE level maskable interrupt[*1] | FEINT | FEINT input | | FE |
| Floating-point arithmetic exception (imprecise) | FPI | Execution of FPU instruction | | EI |
| EI level maskable interrupt[*1] | EIINT | Interrupt controller | | EI |
| Memory protection exception (execution right) | MIP | Memory protection violation | | FE |
| System error exception | SYSERR | Error input at instruction fetch | | FE |
| Reserved instruction exception | RIE | Execution of reserved instruction | | FE |
| Coprocessor unusable exception | UCPOP | Execution of coprocessor instruction/ access right violation | | FE |
| Privileged instruction exception | PIE | Execution of privileged instruction | | FE |
| Misaligned exception | MAE | Generation of misaligned access | | FE |
| Memory protection exception (access right) | MDP | Memory protection violation | | FE |
| Floating-point arithmetic exception (precise) | FPP | Execution of FPU instruction | | EI |
| System call | SYSCALL | Execution of SYSCALL instruction | | EI |
| FE level trap | FETRAP | Execution of FETRAP instruction | | FE |
| EI level trap 0 | TRAP0 | Execution of TRAP instruction | | EI |
| EI level trap 1 | TRAP1 | Execution of TRAP instruction | Low | EI |

Note 1. These interrupt exceptions are described in this section.

### (1) Interrupts

The following three exceptions in **Table 7.1** are called interrupts, and are described in this section.

- FE level non-maskable interrupt (FENMI)
  An FENMI interrupt is acknowledged even if another FE level interrupt - FEINT - has been generated.
  - An FENMI interrupt is acknowledged even if the CPU system register PSW.NP = 1.
  - Return from an FENMI interrupt is not possible and neither is recovery.
- FE level maskable interrupt (FEINT)
  - FEINT can be acknowledged if the CPU system register PSW.NP = 0. It is masked if PSW.NP = 1.
  - Return from an FEINT interrupt is possible and so is recovery.
- EI level maskable interrupt (EIINT)
  An EIINT interrupt can be acknowledged if an FE level interrupt - FENMI or FEINT - has not been generated.
  - EIINT can be acknowledged if the CPU system register PSW.NP = 0.
    It is masked if PSW.NP = 1, EIINT with a higher priority is being processed, or PSW.ID = 1.
  - Return from an EIINT interrupt is possible and so is recovery.
  - Interrupt masking can be specified for each interrupt channel.
  - 16 interrupt priority levels can be specified for each interrupt channel
  - In this section, the EIINT that corresponds to interrupt channel n is indicated by “INTn”, whereas the EIINT that corresponds to interrupt source xxx is indicated by “INTxxx”.

  For the PSW register, see **Table 3.9 PSW Register Contents** and the *RH850G3KH User's Manual: Software*.

**NOTE**

Return: Indicates whether or not the program can resume from where it was interrupted.

Recovery: Indicates whether or not the processor status (status of processor resources including general purpose registers and system registers) can be restored to the status they were in when the program was interrupted.

These interrupt sources are described in **Section 7.2, Interrupt Sources**.

### (2) Overview of interrupts

- Priority levels for interrupt
  16 priority levels of maskable interrupts by request can be set by interrupt control register.
- Detecting methods of external interrupts (TNMI/INTPm)
  A method of detecting external interrupts (TNMI and INTPm) can be selected from five types: rising edge, falling edge, both edges, low level, and high level.
- 2 types of interrupt handler address setting Direct branching method or table referencing method is selectable by register setting.

# 7.2 Interrupt Sources

## 7.2.1 Interrupt Sources

### 7.2.1.1 FE Level Non-Maskable Interrupts

#### (1) Priority

See **Table 7.1, List of Exception Sources**.

#### (2) Return PC

Return or recovery from an FE non-maskable interrupt is not possible.

#### (3) Status Register

See **Section 7.4.4, FNC — FE Level NMI Status Register**.

#### (4) Return Instruction

None

**Table 7.2 FE Level Non-Maskable Interrupt Requests**

| Interrupt | | | Interrupt Request | | Unit | Priority | Exception Source Code | Handler Address 00000... |
|---|---|---|---|---|---|---|---|---|
| | Control Register | | | | | | | |
| Symbol | Name | Address | Name | Source | | | | |
| FENMI | FNC | FFFE EA78$_H$ | TNMI | NMI pin | Port | *1 | 0E0$_H$ | 0E0$_H$ |
| | | | WDTA0NMI | WDTA0 FENMI interrupt | WDTA0 | | | |
| | | | WDTA1NMI | WDTA1 FENMI interrupt | WDTA1 | | | |

Note 1. See **Table 7.1, List of Exception Sources**.

The source of the FENMI interrupt can be evaluated by a dedicated flag register. See **Section 7.2.3, FE Level Non-Maskable Interrupt Sources** for details.

### 7.2.1.2 FE Level Maskable Interrupts

#### (1) Priority

See **Table 7.1, List of Exception Sources**.

#### (2) Return PC

The return PC returned from an interrupt handling routine by the FERET instruction is the PC from when the program was suspended (current PC).

#### (3) Status Register

See **Section 7.4.5, FIC — FE Level Maskable Interrupt Status Register**.

#### (4) Return Instruction

FERET

**Table 7.3 FE Level Maskable Interrupt Requests**

| Interrupt | | | Interrupt Request | | Unit | 100 pin | 144 pin | 176 pin | Priority | Exception Source Code | Handler Address 00000... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Control Register | | | | | | | | | | |
| Symbol | Name | Address | Name | Source | | | | | | | |
| FEINT | FIC | FFFE EA7A$_H$ | INTLVIL | LVI voltage detection (falling) | LVI | √ | √ | √ | *1 | 0F0$_H$ | 0F0$_H$ |
| | | | INTECCDEEP0 | Data flash ECC 1-bit error or 2-bit error interrupt | Data flash | √ | √ | √ | | | |
| | | | INTECCDCNRAM0 | RSCAN0 ECC 1-bit error or 2-bit error interrupt | RSCAN0 (ch0-5) | √ | √ | √ | | | |
| | | | INTECCDCNRAM1 | RSCAN1 ECC 1-bit error or 2-bit error interrupt | RSCAN1 (ch6) | — | — | √ | | | |
| | | | INTECCDCSIH0 | CSIH0 ECC 1-bit error or 2-bit error interrupt | CSIH0 | √ | √ | √ | | | |
| | | | INTECCDCSIH1 | CSIH1 ECC 1-bit error or 2-bit error interrupt | CSIH1 | √ | √ | √ | | | |
| | | | INTECCDCSIH2 | CSIH2 ECC 1-bit error or 2-bit error interrupt | CSIH2 | √ | √ | √ | | | |
| | | | INTECCDCSIH3 | CSIH3 ECC 1-bit error or 2-bit error interrupt | CSIH3 | √ | √ | √ | | | |
| | | | INTECCSCFLI0 | Code Flash ECC 1-bit error or 2-bit error interrupt | Code flash | √ | √ | √ | | | |
| | | | INTECCRAM | RAM ECC 1-bit error or 2-bit error interrupt | RAM | √ | √ | √ | | | |
| | | | INTOSTM0_FE*2 | OSTM0 interrupt | OSTM0 | √ | √ | √ | | | |
| | | | INTLVIH | LVI voltage detection (rising) | LVI | √ | √ | √ | | | |
| | | | INTGUARD | PE Guard (PEG) error or Peripheral Guard (PBG) error or PBG for CPU System error interrupt | CPU, PBG, PBGC | √ | √ | √ | | | |
| | | | INTOSTM1_FE | OSTM1 interrupt | OSTM1 | √ | √ | √ | | | |
| | | | INTOSTM2_FE | OSTM2 interrupt | OSTM2 | √ | √ | √ | | | |
| | | | INTOSTM3_FE | OSTM3 interrupt | OSTM3 | √ | √ | √ | | | |
| | | | INTOSTM4_FE | OSTM4 interrupt | OSTM4 | √ | √ | √ | | | |
| | | | INTDMAERR | DMA transfer error interrupt | DMAC | √ | √ | √ | | | |

Note 1. See **Table 7.1, List of Exception Sources**.

Note 2. INTOSTM0 can operate as an EIINT or FEINT interrupt, but using it both ways at the same time is prohibited.
When INTOSTM0 is used as FEINT, it is generated by the TSU (timing supervision unit) function of OSTM0

* Timing monitor (TSU)
This prevents the illicit use of CPU time by non-trusted programs, manages properties, and controls the intervals over which interrupts are disabled.

### 7.2.1.3 EI Level Maskable Interrupts

#### (1) Interrupt Naming Rules

The composition of the interrupt request signal names, their assigned interrupt control registers and the bits in these registers follow special rules.

In the following the name of the specific interrupt request is represented by **<*name*>**.

For details of the names used in **IC**<*name*>, see **Table 7.4**.

- Interrupt request name: **INT<*name*>**
  The prefix **"INT"** is appended to **<*name*>**.
- Interrupt request control register: **IC<*name*>**
  The prefix **"IC"** is appended to **<*name*>**.
  The 16-bit register **IC<*name*>** can also be accessed in byte units:
  - Low byte (bits [7:0]) of the **IC<*name*>** register: **IC<*name*>L**
    The suffix **"L"** is appended to the register name **IC<*name*>**.
  - High byte (bits [15:8]) of the **IC<*name*>** register: **IC<*name*>H**
    The suffix **"H"** is appended to the register name **IC<*name*>**.
- Interrupt control register bit names: **CT<*name*>**, **RF<*name*>**, **MK<*name*>**, **TB<*name*>**, **P3<*name*>**, **P2<*name*>**, **P1<*name*>**, **P0<*name*>**
  The bit prefix "**CT**", "**RF**", "**MK**", "**TB**", "**P3**", "**P2**", "**P1**", or "**P0**" is appended to the interrupt <*name*>.

**Example**

The interrupt request from channel 2 of TAUD0 channel (**<*name*>** = *TAUD0I2*) is named

**INT***TAUD0I2*

The related interrupt control registers are

**IC***TAUD0I2*, **IC***TAUD0I2L*, **IC***TAUD0I2H*

The bits in this register are

**CT***TAUD0I2*,**RF***TAUD0I2*, **MK***TAUD0I2*, **TB***TAUD0I2*, **P3***TAUD0I2*, **P2***TAUD0I2*, **P1***TAUD0I2*, **P0***TAUD0I2*

#### (2) Priority

See **Table 7.1, List of Exception Sources**.

#### (3) Return PC

The return PC returned from an interrupt handling routine by the EIRET instruction is the PC from when the program was suspended (current PC).

#### (4) Control Register

EI level maskable interrupt control register

See **Section 7.4.2, ICxxx — EI Level Interrupt Control Registers**.

### (5) Return Instruction

EIRET instruction

### (6) Configuration

EI-level maskable interrupts are controlled by the two controllers, INTC1 and INTC2. The interrupts are supported on a total of 358 channels with a cascade connection of INTC1 and INTC2.

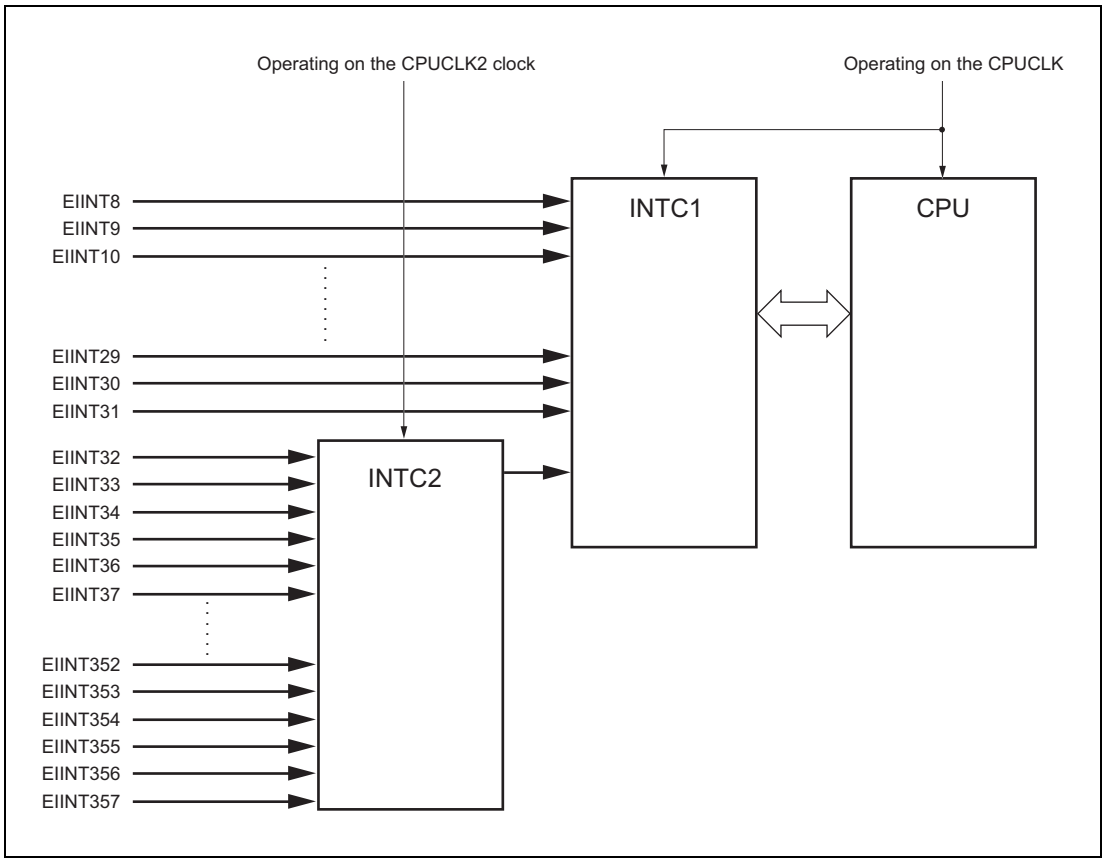



**Figure 7.1 Configuration Diagram of EI Level Maskable Interrupt**

**CAUTION**

**As CPUCLK2 is the operating clock for INTC2, the EIINT32 to EIINT357 interrupts, which are connected to INTC2, are delayed more than the interrupts directly connected to INTC1.**

**Table 7.4** lists EI level maskable interrupts.

Table 7.4 EI Level Maskable Interrupt Sources (1/9)

| Interrupt | | | Interrupt Request | | | | Exception Source Code | 100 pins | 144 pins | 176 pins | Handler Address (Offset)*8 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | Direct Jumping to an Address | | |
| Channel No.*1 | Control Register | | | | | | | | | | | | |
| | Name | Address | Name | Source | Unit | Detection Type*2 | | | | | RINT = 0 | RINT = 1 | Reference to a Table*5 |
| 0 | Reserved | FFFE EA00$_H$ | | | | | 1000$_H$ | — | — | — | *3 | *4 | +000$_H$ |
| 1 | Reserved | FFFE EA02$_H$ | | | | | 1001$_H$ | — | — | — | | | +004$_H$ |
| 2 | Reserved | FFFE EA04$_H$ | | | | | 1002$_H$ | — | — | — | | | +008$_H$ |
| 3 | Reserved | FFFE EA06$_H$ | | | | | 1003$_H$ | — | — | — | | | +00C$_H$ |
| 4 | Reserved | FFFE EA08$_H$ | | | | | 1004$_H$ | — | — | — | | | +010$_H$ |
| 5 | Reserved | FFFE EA0A$_H$ | | | | | 1005$_H$ | — | — | — | | | +014$_H$ |
| 6 | Reserved | FFFE EA0C$_H$ | | | | | 1006$_H$ | — | — | — | | | +018$_H$ |
| 7 | Reserved | FFFE EA0E$_H$ | | | | | 1007$_H$ | — | — | — | | | +01C$_H$ |
| 8 | ICTAUD0I0 | FFFE EA10$_H$ | INTTAUD0I0*6 | Interrupt for CH0 of TAUD0 | TAUD0 | Edge | 1008$_H$ | √ | √ | √ | | | +020$_H$ |
| | ICCSIH2IC_1 | | INTCSIH2IC_1*6 | CSIH2 communication status interrupt | CSIH2 | Edge | | | | | | | |
| 9 | ICTAUD0I2 | FFFE EA12$_H$ | INTTAUD0I2*6 | Interrupt for CH2 of TAUD0 | TAUD0 | Edge | 1009$_H$ | √ | √ | √ | | | +024$_H$ |
| | ICCSIH3IC_1 | | INTCSIH3IC_1*6 | CSIH3 communication status interrupt | CSIH3 | Edge | | | | | | | |
| 10 | ICTAUD0I4 | FFFE EA14$_H$ | INTTAUD0I4*6 | Interrupt for CH4 of TAUD0 | TAUD0 | Edge | 100A$_H$ | √ | √ | √ | | | +028$_H$ |
| | ICCSIH2IR_1 | | INTCSIH2IR_1*6 | CSIH2 receive status interrupt | CSIH2 | Edge | | | | | | | |
| 11 | ICTAUD0I6 | FFFE EA16$_H$ | INTTAUD0I6*6 | Interrupt for CH6 of TAUD0 | TAUD0 | Edge | 100B$_H$ | √ | √ | √ | | | +02C$_H$ |
| | ICCSIH2IRE_1 | | INTCSIH2IRE_1*6 | CSIH2 communication error interrupt | CSIH2 | Edge | | | | | | | |
| 12 | ICTAUD0I8 | FFFE EA18$_H$ | INTTAUD0I8*6 | Interrupt for CH8 of TAUD0 | TAUD0 | Edge | 100C$_H$ | √ | √ | √ | | | +030$_H$ |
| | ICCSIH2IJC_1 | | INTCSIH2IJC_1*6 | CSIH2 job completion interrupt | CSIH2 | Edge | | | | | | | |
| 13 | ICTAUD0I10 | FFFE EA1A$_H$ | INTTAUD0I10*6 | Interrupt for CH10 of TAUD0 | TAUD0 | Edge | 100D$_H$ | √ | √ | √ | | | +034$_H$ |
| | ICCSIH3IR_1 | | INTCSIH3IR_1*6 | CSIH3 receive status interrupt | CSIH3 | Edge | | | | | | | |
| 14 | ICTAUD0I12 | FFFE EA1C$_H$ | INTTAUD0I12*6 | Interrupt for CH12 of TAUD0 | TAUD0 | Edge | 100E$_H$ | √ | √ | √ | | | +038$_H$ |
| | ICCSIH3IRE_1 | | INTCSIH3IRE_1*6 | CSIH3 communication error interrupt | CSIH3 | Edge | | | | | | | |
| 15 | ICTAUD0I14 | FFFE EA1E$_H$ | INTTAUD0I14*6 | Interrupt for CH14 of TAUD0 | TAUD0 | Edge | 100F$_H$ | √ | √ | √ | | | +03C$_H$ |
| | ICCSIH3IJC_1 | | INTCSIH3IJC_1*6 | CSIH3 job completion interrupt | CSIH3 | Edge | | | | | | | |
| 16 | ICTAPA0IPEK0 | FFFE EA20$_H$ | INTTAPA0IPEK0*6 | TAPA0 peak interrupt 0 | TAPA0 | Edge | 1010$_H$ | √ | √ | √ | | | +040$_H$ |
| | ICCSIH1IC_1 | | INTCSIH1IC_1*6 | CSIH1 communication status interrupt | CSIH1 | Edge | | | | | | | |
| 17 | ICTAPA0IVLY0 | FFFE EA22$_H$ | INTTAPA0IVLY0*6 | TAPA0 valley interrupt 0 | TAPA0 | Edge | 1011$_H$ | √ | √ | √ | | | +044$_H$ |
| | ICCSIH1IR_1 | | INTCSIH1IR_1*6 | CSIH1 receive status interrupt | CSIH1 | Edge | | | | | | | |
| 18 | ICADCA0I0 | FFFE EA24$_H$ | INTADCA0I0 | ADCA0 SG1 end interrupt | ADCA0 | Edge | 1012$_H$ | √ | √ | √ | | | +048$_H$ |
| 19 | ICADCA0I1 | FFFE EA26$_H$ | INTADCA0I1 | ADCA0 SG2 end interrupt | ADCA0 | Edge | 1013$_H$ | √ | √ | √ | | | +04C$_H$ |
| 20 | ICADCA0I2 | FFFE EA28$_H$ | INTADCA0I2*6 | ADCA0 SG3 end interrupt | ADCA0 | Edge | 1014$_H$ | √ | √ | √ | | | +050$_H$ |
| | ICCSIH0IJC_1 | | INTCSIH0IJC_1*6 | CSIH0 job completion interrupt | CSIH0 | Edge | | | | | | | |
| 21 | ICDCUTDI | FFFE EA2A$_H$ | INTDCUTDI | Dedicated interrupt for on-chip debug function | Port | Edge | 1015$_H$ | √ | √ | √ | | | +054$_H$ |
| 22 | ICRCANGERR0 | FFFE EA2C$_H$ | INTRCANGERR0 | CAN global error interrupt | RSCAN0 | Level | 1016$_H$ | √ | √ | √ | | | +058$_H$ |
| 23 | ICRCANGRECC0 | FFFE EA2E$_H$ | INTRCANGRECC0 | CAN receive FIFO interrupt | RSCAN0 | Level | 1017$_H$ | √ | √ | √ | | | +05C$_H$ |
| 24 | ICRCAN0ERR | FFFE EA30$_H$ | INTRCAN0ERR | CAN0 error interrupt | RSCAN0 | Level | 1018$_H$ | √ | √ | √ | | | +060$_H$ |
| 25 | ICRCAN0REC | FFFE EA32$_H$ | INTRCAN0REC | CAN0 transmit/receive FIFO receive complete interrupt | RSCAN0 | Level | 1019$_H$ | √ | √ | √ | | | +064$_H$ |
| 26 | ICRCAN0TRX | FFFE EA34$_H$ | INTRCAN0TRX | CAN0 transmit interrupt | RSCAN0 | Level | 101A$_H$ | √ | √ | √ | | | +068$_H$ |
| 27 | ICCSIG0IC | FFFE EA36$_H$ | INTCSIG0IC*6 | CSIG0 communication status interrupt | CSIG0 | Edge | 101B$_H$ | √ | √ | √ | | | +06C$_H$ |
| | ICCSIH1IRE_1 | | INTCSIH1IRE_1*6 | CSIH1 communication error interrupt | CSIH1 | Edge | | | | | | | |
| 28 | ICCSIG0IR | FFFE EA38$_H$ | INTCSIG0IR*6 | CSIG0 receive status interrupt | CSIG0 | Edge | 101C$_H$ | √ | √ | √ | | | +070$_H$ |
| | ICCSIH1IJC_1 | | INTCSIH1IJC_1*6 | CSIH1 job interrupt | CSIH1 | Edge | | | | | | | |
| 29 | ICCSIH0IC | FFFE EA3A$_H$ | INTCSIH0IC | CSIH0 communication status interrupt | CSIH0 | Edge | 101D$_H$ | √ | √ | √ | | | +074$_H$ |

Table 7.4 EI Level Maskable Interrupt Sources (2/9)

| Interrupt | | | Interrupt Request | | | | Exception Source Code | 100 pins | 144 pins | 176 pins | Handler Address (Offset)*8 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | Direct Jumping to an Address | | |
| Channel No.*1 | Control Register | | | | | Detection Type*2 | | | | | | | Reference to a Table*5 |
| | Name | Address | Name | Source | Unit | | | | | | RINT = 0 | RINT = 1 | |
| 30 | ICCSIH0IR | FFFE EA3C$_H$ | INTCSIH0IR | CSIH0 receive status interrupt | CSIH0 | Edge | 101E$_H$ | √ | √ | √ | *3 | *4 | +078$_H$ |
| 31 | ICCSIH0IRE | FFFE EA3E$_H$ | INTCSIH0IRE | CSIH0 communication error interrupt | CSIH0 | Edge | 101F$_H$ | √ | √ | √ | | | +07C$_H$ |
| 32 | ICCSIH0IJC | FFFF B040$_H$ | INTCSIH0IJC*6 | CSIH0 job completion interrupt | CSIH0 | Edge | 1020$_H$ | √ | √ | √ | | | +080$_H$ |
| | ICADCA0I2_2 | | INTADCA0I2_2*6 | ADCA0 SG3 end interrupt | ADCA0 | Edge | | | | | | | |
| 33 | ICRLIN30 | FFFF B042$_H$ | INTRLIN30 | RLIN30 interrupt | RLIN30 | Edge | 1021$_H$ | √ | √ | √ | | | +084$_H$ |
| 34 | ICRLIN30UR0 | FFFF B044$_H$ | INTRLIN30UR0 | RLIN30 transmit interrupt | RLIN30 | Edge | 1022$_H$ | √ | √ | √ | | | +088$_H$ |
| 35 | ICRLIN30UR1 | FFFF B046$_H$ | INTRLIN30UR1 | RLIN30 receive complete interrupt | RLIN30 | Edge | 1023$_H$ | √ | √ | √ | | | +08C$_H$ |
| 36 | ICRLIN30UR2 | FFFF B048$_H$ | INTRLIN30UR2 | RLIN30 status interrupt | RLIN30 | Edge | 1024$_H$ | √ | √ | √ | | | +090$_H$ |
| 37 | ICP0 | FFFF B04A$_H$ | INTP0 | External interrupt | Port | Edge | 1025$_H$ | √ | √ | √ | | | +094$_H$ |
| 38 | ICP1 | FFFF B04C$_H$ | INTP1 | External interrupt | Port | Edge | 1026$_H$ | √ | √ | √ | | | +098$_H$ |
| 39 | ICP2 | FFFF B04E$_H$ | INTP2 | External interrupt | Port | Edge | 1027$_H$ | √ | √ | √ | | | +09C$_H$ |
| 40 | ICWDTA0 | FFFF B050$_H$ | INTWDTA0 | WDTA0 75% interrupt | WDTA0 | Edge | 1028$_H$ | √ | √ | √ | | | +0A0$_H$ |
| 41 | ICWDTA1 | FFFF B052$_H$ | INTWDTA1 | WDTA1 75% interrupt | WDTA1 | Edge | 1029$_H$ | √ | √ | √ | | | +0A4$_H$ |
| 42 | Reserved | FFFF B054$_H$ | | | | | 102A$_H$ | — | — | — | | | +0A8$_H$ |
| 43 | ICP3 | FFFF B056$_H$ | INTP3 | External interrupt | Port | Edge | 102B$_H$ | √ | √ | √ | | | +0AC$_H$ |
| 44 | ICP4 | FFFF B058$_H$ | INTP4 | External interrupt | Port | Edge | 102C$_H$ | √ | √ | √ | | | +0B0$_H$ |
| 45 | ICP5 | FFFF B05A$_H$ | INTP5 | External interrupt | Port | Edge | 102D$_H$ | √ | √ | √ | | | +0B4$_H$ |
| 46 | ICP10 | FFFF B05C$_H$ | INTP10 | External interrupt | Port | Edge | 102E$_H$ | √ | √ | √ | | | +0B8$_H$ |
| 47 | ICP11 | FFFF B05E$_H$ | INTP11 | External interrupt | Port | Edge | 102F$_H$ | √ | √ | √ | | | +0BC$_H$ |
| 48 | ICTAUD0I1 | FFFF B060$_H$ | INTTAUD0I1 | Interrupt for TAUD0 channel 1 | TAUD0 | Edge | 1030$_H$ | √ | √ | √ | | | +0C0$_H$ |
| 49 | ICTAUD0I3 | FFFF B062$_H$ | INTTAUD0I3 | Interrupt for TAUD0 channel 3 | TAUD0 | Edge | 1031$_H$ | √ | √ | √ | | | +0C4$_H$ |
| 50 | ICTAUD0I5 | FFFF B064$_H$ | INTTAUD0I5 | Interrupt for TAUD0 channel 5 | TAUD0 | Edge | 1032$_H$ | √ | √ | √ | | | +0C8$_H$ |
| 51 | ICTAUD0I7 | FFFF B066$_H$ | INTTAUD0I7 | Interrupt for TAUD0 channel 7 | TAUD0 | Edge | 1033$_H$ | √ | √ | √ | | | +0CC$_H$ |
| 52 | ICTAUD0I9 | FFFF B068$_H$ | INTTAUD0I9 | Interrupt for TAUD0 channel 9 | TAUD0 | Edge | 1034$_H$ | √ | √ | √ | | | +0D0$_H$ |
| 53 | ICTAUD0I11 | FFFF B06A$_H$ | INTTAUD0I11 | Interrupt for TAUD0 channel 11 | TAUD0 | Edge | 1035$_H$ | √ | √ | √ | | | +0D4$_H$ |
| 54 | ICTAUD0I13 | FFFF B06C$_H$ | INTTAUD0I13 | Interrupt for TAUD0 channel 13 | TAUD0 | Edge | 1036$_H$ | √ | √ | √ | | | +0D8$_H$ |
| 55 | ICTAUD0I15 | FFFF B06E$_H$ | INTTAUD0I15 | Interrupt for TAUD0 channel 15 | TAUD0 | Edge | 1037$_H$ | √ | √ | √ | | | +0DC$_H$ |
| 56 | ICADCA0ERR | FFFF B070$_H$ | INTADCA0ERR | ADCA0 error interrupt | ADCA0 | Edge | 1038$_H$ | √ | √ | √ | | | +0E0$_H$ |
| 57 | ICCSIG0IRE | FFFF B072$_H$ | INTCSIG0IRE | CSIG0 receive error interrupt | CSIG0 | Edge | 1039$_H$ | √ | √ | √ | | | +0E4$_H$ |
| 58 | ICRLIN20 | FFFF B074$_H$ | INTRLIN20 | RLIN20 interrupt | RLIN240 | Edge | 103A$_H$ | √ | √ | √ | | | +0E8$_H$ |
| 59 | ICRLIN21 | FFFF B076$_H$ | INTRLIN21 | RLIN21 interrupt | RLIN240 | Edge | 103B$_H$ | √ | √ | √ | | | +0EC$_H$ |
| 60 | ICDMA0 | FFFF B078$_H$ | INTDMA0 | DMA0 transfer completion | DMAC | Edge | 103C$_H$ | √ | √ | √ | | | +0F0$_H$ |
| 61 | ICDMA1 | FFFF B07A$_H$ | INTDMA1 | DMA1 transfer completion | DMAC | Edge | 103D$_H$ | √ | √ | √ | | | +0F4$_H$ |
| 62 | ICDMA2 | FFFF B07C$_H$ | INTDMA2 | DMA2 transfer completion | DMAC | Edge | 103E$_H$ | √ | √ | √ | | | +0F8$_H$ |
| 63 | ICDMA3 | FFFF B07E$_H$ | INTDMA3 | DMA3 transfer completion | DMAC | Edge | 103F$_H$ | √ | √ | √ | | | +0FC$_H$ |
| 64 | ICDMA4 | FFFF B080$_H$ | INTDMA4 | DMA4 transfer completion | DMAC | Edge | 1040$_H$ | √ | √ | √ | | | +100$_H$ |
| 65 | ICDMA5 | FFFF B082$_H$ | INTDMA5 | DMA5 transfer completion | DMAC | Edge | 1041$_H$ | √ | √ | √ | | | +104$_H$ |
| 66 | ICDMA6 | FFFF B084$_H$ | INTDMA6 | DMA6 transfer completion | DMAC | Edge | 1042$_H$ | √ | √ | √ | | | +108$_H$ |
| 67 | ICDMA7 | FFFF B086$_H$ | INTDMA7 | DMA7 transfer completion | DMAC | Edge | 1043$_H$ | √ | √ | √ | | | +10C$_H$ |
| 68 | ICDMA8 | FFFF B088$_H$ | INTDMA8 | DMA8 transfer completion | DMAC | Edge | 1044$_H$ | √ | √ | √ | | | +110$_H$ |
| 69 | ICDMA9 | FFFF B08A$_H$ | INTDMA9 | DMA9 transfer completion | DMAC | Edge | 1045$_H$ | √ | √ | √ | | | +114$_H$ |
| 70 | ICDMA10 | FFFF B08C$_H$ | INTDMA10 | DMA10 transfer completion | DMAC | Edge | 1046$_H$ | √ | √ | √ | | | +118$_H$ |
| 71 | ICDMA11 | FFFF B08E$_H$ | INTDMA11 | DMA11 transfer completion | DMAC | Edge | 1047$_H$ | √ | √ | √ | | | +11C$_H$ |
| 72 | ICDMA12 | FFFF B090$_H$ | INTDMA12 | DMA12 transfer completion | DMAC | Edge | 1048$_H$ | √ | √ | √ | | | +120$_H$ |
| 73 | ICDMA13 | FFFF B092$_H$ | INTDMA13 | DMA13 transfer completion | DMAC | Edge | 1049$_H$ | √ | √ | √ | | | +124$_H$ |
| 74 | ICDMA14 | FFFF B094$_H$ | INTDMA14 | DMA14 transfer completion | DMAC | Edge | 104A$_H$ | √ | √ | √ | | | +128$_H$ |

Table 7.4 EI Level Maskable Interrupt Sources (3/9)

| Interrupt | | | Interrupt Request | | | | Exception Source Code | 100 pins | 144 pins | 176 pins | Handler Address (Offset)*8 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | Direct Jumping to an Address | | |
| Channel No.*1 | Control Register | | | | | | | | | | | | |
| | Name | Address | Name | Source | Unit | Detection Type*2 | | | | | RINT = 0 | RINT = 1 | Reference to a Table*5 |
| 75 | ICDMA15 | FFFF B096$_H$ | INTDMA15 | DMA15 transfer completion | DMAC | Edge | 104B$_H$ | √ | √ | √ | *3 | *4 | +12C$_H$ |
| 76 | ICRIIC0TI | FFFF B098$_H$ | INTRIIC0TI | RIIC transmit data empty interrupt | RIIC0 | Edge | 104C$_H$ | √ | √ | √ | | | +130$_H$ |
| 77 | ICRIIC0EE | FFFF B09A$_H$ | INTRIIC0EE | RIIC receive error/event interrupt | RIIC0 | Level | 104D$_H$ | √ | √ | √ | | | +134$_H$ |
| 78 | ICRIIC0RI | FFFF B09C$_H$ | INTRIIC0RI | RIIC receive complete interrupt | RIIC0 | Edge | 104E$_H$ | √ | √ | √ | | | +138$_H$ |
| 79 | ICRIIC0TEI | FFFF B09E$_H$ | INTRIIC0TEI | RIIC transmit complete interrupt | RIIC0 | Level | 104F$_H$ | √ | √ | √ | | | +13C$_H$ |
| 80 | ICTAUJ0I0 | FFFF B0A0$_H$ | INTTAUJ0I0 | Interrupt for TAUJ0 channel 0 | TAUJ0 | Edge | 1050$_H$ | √ | √ | √ | | | +140$_H$ |
| 81 | ICTAUJ0I1 | FFFF B0A2$_H$ | INTTAUJ0I1 | Interrupt for TAUJ0 channel 1 | TAUJ0 | Edge | 1051$_H$ | √ | √ | √ | | | +144$_H$ |
| 82 | ICTAUJ0I2 | FFFF B0A4$_H$ | INTTAUJ0I2 | Interrupt for TAUJ0 channel 2 | TAUJ0 | Edge | 1052$_H$ | √ | √ | √ | | | +148$_H$ |
| 83 | ICTAUJ0I3 | FFFF B0A6$_H$ | INTTAUJ0I3 | Interrupt for TAUJ0 channel 3 | TAUJ0 | Edge | 1053$_H$ | √ | √ | √ | | | +14C$_H$ |
| 84 | ICOSTM0 | FFFF B0A8$_H$ | INTOSTM0*9 | OSTM0 interrupt | OSTM0 | Edge | 1054$_H$ | √ | √ | √ | | | +150$_H$ |
| 85 | ICENCA0IOV | FFFF B0AA$_H$ | INTENCA0IOV*7 | ENCA0 overflow interrupt | ENCA0 | Edge | 1055$_H$ | √ | √ | √ | | | +154$_H$ |
| | ICPWGA4 | | INTPWGA4*7 | PWGA4 interrupt | PWGA4 | Edge | | | | | | | |
| 86 | ICENCA0IUD | FFFF B0AC$_H$ | INTENCA0IUD*7 | ENCA0 underflow interrupt | ENCA0 | Edge | 1056$_H$ | √ | √ | √ | | | +158$_H$ |
| | ICPWGA5 | | INTPWGA5*7 | PWGA5 interrupt | PWGA5 | Edge | | | | | | | |
| 87 | ICENCA0I0 | FFFF B0AE$_H$ | INTENCA0I0*7 | ENCA0 match/capture interrupt 0 | ENCA0 | Edge | 1057$_H$ | √ | √ | √ | | | +15C$_H$ |
| | ICPWGA6 | | INTPWGA6*7 | PWGA6 interrupt | PWGA6 | Edge | | | | | | | |
| 88 | ICENCA0I1 | FFFF B0B0$_H$ | INTENCA0I1*7 | ENCA0 match/capture interrupt 1 | ENCA0 | Edge | 1058$_H$ | √ | √ | √ | | | +160$_H$ |
| | ICPWGA7 | | INTPWGA7*7 | PWGA7 interrupt | PWGA7 | Edge | | | | | | | |
| 89 | ICENCA0IEC | FFFF B0B2$_H$ | INTENCA0IEC | ENCA0 encoder clear interrupt | ENCA0 | Edge | 1059$_H$ | √ | √ | √ | | | +164$_H$ |
| 90 | ICKR0 | FFFF B0B4$_H$ | INTKR0 | KR0 key interrupt | KR0 | Edge | 105A$_H$ | √ | √ | √ | | | +168$_H$ |
| 91 | ICQFULL | FFFF B0B6$_H$ | INTQFULL | PWSA queue full interrupt | PWSA0 | Edge | 105B$_H$ | √ | √ | √ | | | +16C$_H$ |
| 92 | ICPWGA0 | FFFF B0B8$_H$ | INTPWGA0 | PWGA0 interrupt | PWGA0 | Edge | 105C$_H$ | √ | √ | √ | | | +170$_H$ |
| 93 | ICPWGA1 | FFFF B0BA$_H$ | INTPWGA1 | PWGA1 interrupt | PWGA1 | Edge | 105D$_H$ | √ | √ | √ | | | +174$_H$ |
| 94 | ICPWGA2 | FFFF B0BC$_H$ | INTPWGA2 | PWGA2 interrupt | PWGA2 | Edge | 105E$_H$ | √ | √ | √ | | | +178$_H$ |
| 95 | ICPWGA3 | FFFF B0BE$_H$ | INTPWGA3 | PWGA3 interrupt | PWGA3 | Edge | 105F$_H$ | √ | √ | √ | | | +17C$_H$ |
| 96 | ICPWGA8 | FFFF B0C0$_H$ | INTPWGA8 | PWGA8 interrupt | PWGA8 | Edge | 1060$_H$ | √ | √ | √ | | | +180$_H$ |
| 97 | ICPWGA9 | FFFF B0C2$_H$ | INTPWGA9 | PWGA9 interrupt | PWGA9 | Edge | 1061$_H$ | √ | √ | √ | | | +184$_H$ |
| 98 | ICPWGA10 | FFFF B0C4$_H$ | INTPWGA10 | PWGA10 interrupt | PWGA10 | Edge | 1062$_H$ | √ | √ | √ | | | +188$_H$ |
| 99 | ICPWGA11 | FFFF B0C6$_H$ | INTPWGA11 | PWGA11 interrupt | PWGA11 | Edge | 1063$_H$ | √ | √ | √ | | | +18C$_H$ |
| 100 | ICPWGA12 | FFFF B0C8$_H$ | INTPWGA12 | PWGA12 interrupt | PWGA12 | Edge | 1064$_H$ | √ | √ | √ | | | +190$_H$ |
| 101 | ICPWGA13 | FFFF B0CA$_H$ | INTPWGA13 | PWGA13 interrupt | PWGA13 | Edge | 1065$_H$ | √ | √ | √ | | | +194$_H$ |
| 102 | ICPWGA14 | FFFF B0CC$_H$ | INTPWGA14 | PWGA14 interrupt | PWGA14 | Edge | 1066$_H$ | √ | √ | √ | | | +198$_H$ |
| 103 | ICPWGA15 | FFFF B0CE$_H$ | INTPWGA15 | PWGA15 interrupt | PWGA15 | Edge | 1067$_H$ | √ | √ | √ | | | +19C$_H$ |
| 104 | Reserved | FFFF B0D0$_H$ | | | | | 1068$_H$ | — | — | — | | | +1A0$_H$ |
| 105 | Reserved | FFFF B0D2$_H$ | | | | | 1069$_H$ | — | — | — | | | +1A4$_H$ |
| 106 | Reserved | FFFF B0D4$_H$ | | | | | 106A$_H$ | — | — | — | | | +1A8$_H$ |
| 107 | Reserved | FFFF B0D6$_H$ | | | | | 106B$_H$ | — | — | — | | | +1AC$_H$ |
| 108 | Reserved | FFFF B0D8$_H$ | | | | | 106C$_H$ | — | — | — | | | +1B0$_H$ |
| 109 | Reserved | FFFF B0DA$_H$ | | | | | 106D$_H$ | — | — | — | | | +1B4$_H$ |
| 110 | ICFLERR | FFFF B0DC$_H$ | INTFLERR*10 | Flash sequencer end error interrupt | FACI | Level | 106E$_H$ | √ | √ | √ | | | +1B8$_H$ |
| 111 | ICFLENDNM | FFFF B0DE$_H$ | INTFLENDNM*10 | Flash sequencer end interrupt | FACI | Edge | 106F$_H$ | √ | √ | √ | | | +1BC$_H$ |
| 112 | ICCWEND | FFFF B0E0$_H$ | INTCWEND | LPS port polling end interrupt | LPS0 | Edge | 1070$_H$ | √ | √ | √ | | | +1C0$_H$ |
| 113 | ICRCAN1ERR | FFFF B0E2$_H$ | INTRCAN1ERR | CAN1 error interrupt | RSCAN0 | Level | 1071$_H$ | √ | √ | √ | | | +1C4$_H$ |
| 114 | ICRCAN1REC | FFFF B0E4$_H$ | INTRCAN1REC | CAN1 transmit/receive FIFO receive complete interrupt | RSCAN0 | Level | 1072$_H$ | √ | √ | √ | | | +1C8$_H$ |
| 115 | ICRCAN1TRX | FFFF B0E6$_H$ | INTRCAN1TRX | CAN1 transmit interrupt | RSCAN0 | Level | 1073$_H$ | √ | √ | √ | | | +1CC$_H$ |

## Table 7.4 EI Level Maskable Interrupt Sources (4/9)

| Interrupt | | | Interrupt Request | | | | Exception Source Code | 100 pins | 144 pins | 176 pins | Handler Address (Offset)*8 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Control Register | | | | | | | | | | Direct Jumping to an Address | | |
| Channel No.*1 | Name | Address | Name | Source | Unit | Detection Type*2 | | | | | RINT = 0 | RINT = 1 | Reference to a Table*5 |
| 116 | ICCSIH1IC | FFFF B0E8$_H$ | INTCSIH1IC*6 | CSIH1 communication status interrupt | CSIH1 | Edge | 1074$_H$ | √ | √ | √ | *3 | *4 | +1D0$_H$ |
| | ICTAPA0IPEK0_2 | | INTTAPA0IPEK0_2*6 | TAPA0 peak interrupt 0 | TAPA0 | Edge | | | | | | | |
| 117 | ICCSIH1IR | FFFF B0EA$_H$ | INTCSIH1IR*6 | CSIH1 receive status interrupt | CSIH1 | Edge | 1075$_H$ | √ | √ | √ | | | +1D4$_H$ |
| | ICTAPA0IVLY0_2 | | INTTAPA0IVLY0_2*6 | TAPA0 valley interrupt 0 | TAPA0 | Edge | | | | | | | |
| 118 | ICCSIH1IRE | FFFF B0EC$_H$ | INTCSIH1IRE*6 | CSIH1 communication error interrupt | CSIH1 | Edge | 1076$_H$ | √ | √ | √ | | | +1D8$_H$ |
| | ICCSIG0IC_2 | | INTCSIG0IC_2*6 | CSIG0 communication status interrupt | CSIG0 | Edge | | | | | | | |
| 119 | ICCSIH1IJC | FFFF B0EE$_H$ | INTCSIH1IJC*6 | CSIH1 job interrupt | CSIH1 | Edge | 1077$_H$ | √ | √ | √ | | | +1DC$_H$ |
| | ICCSIG0IR_2 | | INTCSIG0IR_2*6 | CSIG0 receive status interrupt | CSIG0 | Edge | | | | | | | |
| 120 | ICRLIN31 | FFFF B0F0$_H$ | INTRLIN31 | RLIN31 interrupt | RLIN31 | Edge | 1078$_H$ | √ | √ | √ | | | +1E0$_H$ |
| 121 | ICRLIN31UR0 | FFFF B0F2$_H$ | INTRLIN31UR0 | RLIN31 transmit interrupt | RLIN31 | Edge | 1079$_H$ | √ | √ | √ | | | +1E4$_H$ |
| 122 | ICRLIN31UR1 | FFFF B0F4$_H$ | INTRLIN31UR1 | RLIN31 receive complete interrupt | RLIN31 | Edge | 107A$_H$ | √ | √ | √ | | | +1E8$_H$ |
| 123 | ICRLIN31UR2 | FFFF B0F6$_H$ | INTRLIN31UR2 | RLIN31 status interrupt | RLIN31 | Edge | 107B$_H$ | √ | √ | √ | | | +1EC$_H$ |
| 124 | ICPWGA20 | FFFF B0F8$_H$ | INTPWGA20 | PWGA20 interrupt | PWGA20 | Edge | 107C$_H$ | √ | √ | √ | | | +1F0$_H$ |
| 125 | ICPWGA21 | FFFF B0FA$_H$ | INTPWGA21 | PWGA21 interrupt | PWGA21 | Edge | 107D$_H$ | √ | √ | √ | | | +1F4$_H$ |
| 126 | ICPWGA22 | FFFF B0FC$_H$ | INTPWGA22 | PWGA22 interrupt | PWGA22 | Edge | 107E$_H$ | √ | √ | √ | | | +1F8$_H$ |
| 127 | ICPWGA23 | FFFF B0FE$_H$ | INTPWGA23 | PWGA23 interrupt | PWGA23 | Edge | 107F$_H$ | √ | √ | √ | | | +1FC$_H$ |
| 128 | ICP6 | FFFF B100$_H$ | INTP6 | External interrupt | Port | Edge | 1080$_H$ | √ | √ | √ | | | +200$_H$ |
| 129 | ICP7 | FFFF B102$_H$ | INTP7 | External interrupt | Port | Edge | 1081$_H$ | √ | √ | √ | | | +204$_H$ |
| 130 | ICP8 | FFFF B104$_H$ | INTP8 | External interrupt | Port | Edge | 1082$_H$ | √ | √ | √ | | | +208$_H$ |
| 131 | ICP12 | FFFF B106$_H$ | INTP12 | External interrupt | Port | Edge | 1083$_H$ | √ | √ | √ | | | +20C$_H$ |
| 132 | ICCSIH2IC | FFFF B108$_H$ | INTCSIH2IC*6 | CSIH2 communication status interrupt | CSIH2 | Edge | 1084$_H$ | √ | √ | √ | | | +210$_H$ |
| | ICTAUD0I0_2 | | INTTAUD0I0_2*6 | Interrupt for TAUD0 channel 0 | TAUD0 | Edge | | | | | | | |
| 133 | ICCSIH2IR | FFFF B10A$_H$ | INTCSIH2IR*6 | CSIH2 communication receive interrupt | CSIH2 | Edge | 1085$_H$ | √ | √ | √ | | | +214$_H$ |
| | ICTAUD0I4_2 | | INTTAUD0I4_2*6 | Interrupt for TAUD0 channel 4 | TAUD0 | Edge | | | | | | | |
| 134 | ICCSIH2IRE | FFFF B10C$_H$ | INTCSIH2IRE*6 | CSIH2 communication error interrupt | CSIH2 | Edge | 1086$_H$ | √ | √ | √ | | | +218$_H$ |
| | ICTAUD0I6_2 | | INTTAUD0I6_2*6 | Interrupt for TAUD0 channel 6 | TAUD0 | Edge | | | | | | | |
| 135 | ICCSIH2IJC | FFFF B10E$_H$ | INTCSIH2IJC*6 | CSIH2 job completion interrupt | CSIH2 | Edge | 1087$_H$ | √ | √ | √ | | | +21C$_H$ |
| | ICTAUD0I8_2 | | INTTAUD0I8_2*6 | Interrupt for TAUD0 channel 8 | TAUD0 | Edge | | | | | | | |
| 136 | Reserved | FFFF B110$_H$ | | | | | 1088$_H$ | — | — | — | | | +220$_H$ |
| 137 | Reserved | FFFF B112$_H$ | | | | | 1089$_H$ | — | — | — | | | +224$_H$ |
| 138 | Reserved | FFFF B114$_H$ | | | | | 108A$_H$ | — | — | — | | | +228$_H$ |
| 139 | Reserved | FFFF B116$_H$ | | | | | 108B$_H$ | — | — | — | | | +22C$_H$ |
| 140 | Reserved | FFFF B118$_H$ | | | | | 108C$_H$ | — | — | — | | | +230$_H$ |
| 141 | Reserved | FFFF B11A$_H$ | | | | | 108D$_H$ | — | — | — | | | +234$_H$ |
| 142 | ICTAUB0I0 | FFFF B11C$_H$ | INTTAUB0I0 | Interrupt for TAUB0 channel 0 | TAUB0 | Edge | 108E$_H$ | √ | √ | √ | | | +238$_H$ |
| 143 | ICTAUB0I1 | FFFF B11E$_H$ | INTTAUB0I1 | Interrupt for TAUB0 channel 1 | TAUB0 | Edge | 108F$_H$ | √ | √ | √ | | | +23C$_H$ |
| 144 | ICTAUB0I2 | FFFF B120$_H$ | INTTAUB0I2 | Interrupt for TAUB0 channel 2 | TAUB0 | Edge | 1090$_H$ | √ | √ | √ | | | +240$_H$ |
| 145 | ICTAUB0I3 | FFFF B122$_H$ | INTTAUB0I3*7 | Interrupt for TAUB0 channel 3 | TAUB0 | Edge | 1091$_H$ | √ | √ | √ | | | +244$_H$ |
| | ICPWGA16 | | INTPWGA16*7 | PWGA16 interrupt | PWGA16 | Edge | | | | | | | |
| 146 | ICTAUB0I4 | FFFF B124$_H$ | INTTAUB0I4 | Interrupt for TAUB0 channel 4 | TAUB0 | Edge | 1092$_H$ | √ | √ | √ | | | +248$_H$ |
| 147 | ICTAUB0I5 | FFFF B126$_H$ | INTTAUB0I5*7 | Interrupt for TAUB0 channel 5 | TAUB0 | Edge | 1093$_H$ | √ | √ | √ | | | +24C$_H$ |
| | ICPWGA17 | | INTPWGA17*7 | PWGA17 interrupt | PWGA17 | Edge | | | | | | | |
| 148 | ICTAUB0I6 | FFFF B128$_H$ | INTTAUB0I6 | Interrupt for TAUB0 channel 6 | TAUB0 | Edge | 1094$_H$ | √ | √ | √ | | | +250$_H$ |
| 149 | ICTAUB0I7 | FFFF B12A$_H$ | INTTAUB0I7*7 | Interrupt for TAUB0 channel 7 | TAUB0 | Edge | 1095$_H$ | √ | √ | √ | | | +254$_H$ |
| | ICPWGA18 | | INTPWGA18*7 | PWGA18 interrupt | PWGA18 | Edge | | | | | | | |
| 150 | ICTAUB0I8 | FFFF B12C$_H$ | INTTAUB0I8 | Interrupt for TAUB0 channel 8 | TAUB0 | Edge | 1096$_H$ | √ | √ | √ | | | +258$_H$ |

Table 7.4 EI Level Maskable Interrupt Sources (5/9)

| Interrupt | | | Interrupt Request | | | | Exception Source Code | 100 pins | 144 pins | 176 pins | Handler Address (Offset)*8 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | Direct Jumping to an Address | | |
| Channel No.*1 | Control Register | | | | | | | | | | | | |
| | Name | Address | Name | Source | Unit | Detection Type*2 | | | | | RINT = 0 | RINT = 1 | Reference to a Table*5 |
| 151 | ICTAUB0I9 | FFFF B12E$_H$ | INTTAUB0I9*7 | Interrupt for TAUB0 channel 9 | TAUB0 | Edge | 1097$_H$ | √ | √ | √ | *3 | *4 | +25C$_H$ |
| | ICPWGA19 | | INTPWGA19*7 | PWGA19 interrupt | PWGA19 | Edge | | | | | | | |
| 152 | ICTAUB0I10 | FFFF B130$_H$ | INTTAUB0I10 | Interrupt for TAUB0 channel 10 | TAUB0 | Edge | 1098$_H$ | √ | √ | √ | | | +260$_H$ |
| 153 | ICTAUB0I11 | FFFF B132$_H$ | INTTAUB0I11*7 | Interrupt for TAUB0 channel 11 | TAUB0 | Edge | 1099$_H$ | √ | √ | √ | | | +264$_H$ |
| | ICPWGA26 | | INTPWGA26*7 | PWGA26 interrupt | PWGA26 | Edge | | | | | | | |
| 154 | ICTAUB0I12 | FFFF B134$_H$ | INTTAUB0I12 | Interrupt for TAUB0 channel 12 | TAUB0 | Edge | 109A$_H$ | √ | √ | √ | | | +268$_H$ |
| 155 | ICTAUB0I13 | FFFF B136$_H$ | INTTAUB0I13*7 | Interrupt for TAUB0 channel 13 | TAUB0 | Edge | 109B$_H$ | √ | √ | √ | | | +26C$_H$ |
| | ICPWGA30 | | INTPWGA30*7 | PWGA30 interrupt | PWGA30 | Edge | | | | | | | |
| 156 | ICTAUB0I14 | FFFF B138$_H$ | INTTAUB0I14 | Interrupt for TAUB0 channel 14 | TAUB0 | Edge | 109C$_H$ | √ | √ | √ | | | +270$_H$ |
| 157 | ICTAUB0I15 | FFFF B13A$_H$ | INTTAUB0I15*7 | Interrupt for TAUB0 channel 15 | TAUB0 | Edge | 109D$_H$ | √ | √ | √ | | | +274$_H$ |
| | ICPWGA31 | | INTPWGA31*7 | PWGA31 interrupt | PWGA31 | Edge | | | | | | | |
| 158 | ICCSIH3IC | FFFF B13C$_H$ | INTCSIH3IC*6 | CSIH3 communication status interrupt | CSIH3 | Edge | 109E$_H$ | √ | √ | √ | | | +278$_H$ |
| | ICTAUD0I2_2 | | INTTAUD0I2_2*6 | Interrupt for TAUD0 channel 2 | TAUD0 | Edge | | | | | | | |
| 159 | ICCSIH3IR | FFFF B13E$_H$ | INTCSIH3IR*6 | CSIH3 receive status interrupt | CSIH3 | Edge | 109F$_H$ | √ | √ | √ | | | +27C$_H$ |
| | ICTAUD0I10_2 | | INTTAUD0I10_2*6 | Interrupt for TAUD0 channel 10 | TAUD0 | Edge | | | | | | | |
| 160 | ICCSIH3IRE | FFFF B140$_H$ | INTCSIH3IRE*6 | CSIH3 communication error interrupt | CSIH3 | Edge | 10A0$_H$ | √ | √ | √ | | | +280$_H$ |
| | ICTAUD0I12_2 | | INTTAUD0I12_2*6 | Interrupt for TAUD0 channel 12 | TAUD0 | Edge | | | | | | | |
| 161 | ICCSIH3IJC | FFFF B142$_H$ | INTCSIH3IJC*6 | CSIH3 job completion interrupt | CSIH3 | Edge | 10A1$_H$ | √ | √ | √ | | | +284$_H$ |
| | ICTAUD0I14_2 | | INTTAUD0I14_2*6 | Interrupt for TAUD0 channel 14 | TAUD0 | Edge | | | | | | | |
| 162 | ICRLIN22 | FFFF B144$_H$ | INTRLIN22 | RLIN22 interrupt | RLIN240 | Edge | 10A2$_H$ | √ | √ | √ | | | +288$_H$ |
| 163 | ICRLIN23 | FFFF B146$_H$ | INTRLIN23 | RLIN23 interrupt | RLIN240 | Edge | 10A3$_H$ | — | √ | √ | | | +28C$_H$ |
| 164 | ICRLIN32 | FFFF B148$_H$ | INTRLIN32 | RLIN32 interrupt | RLIN32 | Edge | 10A4$_H$ | √ | √ | √ | | | +290$_H$ |
| 165 | ICRLIN32UR0 | FFFF B14A$_H$ | INTRLIN32UR0 | RLIN32 transmit interrupt | RLIN32 | Edge | 10A5$_H$ | √ | √ | √ | | | +294$_H$ |
| 166 | ICRLIN32UR1 | FFFF B14C$_H$ | INTRLIN32UR1 | RLIN32 receive complete interrupt | RLIN32 | Edge | 10A6$_H$ | √ | √ | √ | | | +298$_H$ |
| 167 | ICRLIN32UR2 | FFFF B14E$_H$ | INTRLIN32UR2 | RLIN32 status interrupt | RLIN32 | Edge | 10A7$_H$ | √ | √ | √ | | | +29C$_H$ |
| 168 | ICTAUJ1I0 | FFFF B150$_H$ | INTTAUJ1I0 | Interrupt for TAUJ1 channel 0 | TAUJ1 | Edge | 10A8$_H$ | √ | √ | √ | | | +2A0$_H$ |
| 169 | ICTAUJ1I1 | FFFF B152$_H$ | INTTAUJ1I1 | Interrupt for TAUJ1 channel 1 | TAUJ1 | Edge | 10A9$_H$ | √ | √ | √ | | | +2A4$_H$ |
| 170 | ICTAUJ1I2 | FFFF B154$_H$ | INTTAUJ1I2 | Interrupt for TAUJ1 channel 2 | TAUJ1 | Edge | 10AA$_H$ | √ | √ | √ | | | +2A8$_H$ |
| 171 | ICTAUJ1I3 | FFFF B156$_H$ | INTTAUJ1I3 | Interrupt for TAUJ1 channel 3 | TAUJ1 | Edge | 10AB$_H$ | √ | √ | √ | | | +2AC$_H$ |
| 172 | Reserved | FFFF B158$_H$ | | | | | 10AC$_H$ | — | — | — | | | +2B0$_H$ |
| 173 | Reserved | FFFF B15A$_H$ | | | | | 10AD$_H$ | — | — | — | | | +2B4$_H$ |
| 174 | Reserved | FFFF B15C$_H$ | | | | | 10AE$_H$ | — | — | — | | | +2B8$_H$ |
| 175 | Reserved | FFFF B15E$_H$ | | | | | 10AF$_H$ | — | — | — | | | +2BC$_H$ |
| 176 | Reserved | FFFF B160$_H$ | | | | | 10B0$_H$ | — | — | — | | | +2C0$_H$ |
| 177 | Reserved | FFFF B162$_H$ | | | | | 10B1$_H$ | — | — | — | | | +2C4$_H$ |
| 178 | Reserved | FFFF B164$_H$ | | | | | 10B2$_H$ | — | — | — | | | +2C8$_H$ |
| 179 | Reserved | FFFF B166$_H$ | | | | | 10B3$_H$ | — | — | — | | | +2CC$_H$ |
| 180 | Reserved | FFFF B168$_H$ | | | | | 10B4$_H$ | — | — | — | | | +2D0$_H$ |
| 181 | Reserved | FFFF B16A$_H$ | | | | | 10B5$_H$ | — | — | — | | | +2D4$_H$ |
| 182 | Reserved | FFFF B16C$_H$ | | | | | 10B6$_H$ | — | — | — | | | +2D8$_H$ |
| 183 | Reserved | FFFF B16E$_H$ | | | | | 10B7$_H$ | — | — | — | | | +2DC$_H$ |
| 184 | ICPWGA24 | FFFF B170$_H$ | INTPWGA24 | PWGA24 interrupt | PWGA24 | Edge | 10B8$_H$ | √ | √ | √ | | | +2E0$_H$ |
| 185 | ICPWGA25 | FFFF B172$_H$ | INTPWGA25 | PWGA25 interrupt | PWGA25 | Edge | 10B9$_H$ | √ | √ | √ | | | +2E4$_H$ |
| 186 | ICPWGA27 | FFFF B174$_H$ | INTPWGA27 | PWGA27 interrupt | PWGA27 | Edge | 10BA$_H$ | √ | √ | √ | | | +2E8$_H$ |
| 187 | ICPWGA28 | FFFF B176$_H$ | INTPWGA28 | PWGA28 interrupt | PWGA28 | Edge | 10BB$_H$ | √ | √ | √ | | | +2EC$_H$ |
| 188 | ICPWGA29 | FFFF B178$_H$ | INTPWGA29 | PWGA29 interrupt | PWGA29 | Edge | 10BC$_H$ | √ | √ | √ | | | +2F0$_H$ |

Table 7.4 EI Level Maskable Interrupt Sources (6/9)

| Interrupt | | | Interrupt Request | | | | Exception Source Code | 100 pins | 144 pins | 176 pins | Handler Address (Offset)*8 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Channel No.*1 | Control Register | | Name | Source | Unit | Detection Type*2 | | | | | Direct Jumping to an Address | | Reference to a Table*5 |
| | Name | Address | | | | | | | | | RINT = 0 | RINT = 1 | |
| 189 | ICPWGA32 | FFFF B17A$_H$ | INTPWGA32 | PWGA32 interrupt | PWGA32 | Edge | 10BD$_H$ | √ | √ | √ | *3 | *4 | +2F4$_H$ |
| 190 | ICPWGA33 | FFFF B17C$_H$ | INTPWGA33 | PWGA33 interrupt | PWGA33 | Edge | 10BE$_H$ | √ | √ | √ | | | +2F8$_H$ |
| 191 | ICPWGA34 | FFFF B17E$_H$ | INTPWGA34 | PWGA34 interrupt | PWGA34 | Edge | 10BF$_H$ | √ | √ | √ | | | +2FC$_H$ |
| 192 | ICPWGA35 | FFFF B180$_H$ | INTPWGA35 | PWGA35 interrupt | PWGA35 | Edge | 10C0$_H$ | √ | √ | √ | | | +300$_H$ |
| 193 | ICPWGA36 | FFFF B182$_H$ | INTPWGA36 | PWGA36 interrupt | PWGA36 | Edge | 10C1$_H$ | √ | √ | √ | | | +304$_H$ |
| 194 | ICPWGA37 | FFFF B184$_H$ | INTPWGA37 | PWGA37 interrupt | PWGA37 | Edge | 10C2$_H$ | √ | √ | √ | | | +308$_H$ |
| 195 | ICPWGA38 | FFFF B186$_H$ | INTPWGA38 | PWGA38 interrupt | PWGA38 | Edge | 10C3$_H$ | √ | √ | √ | | | +30C$_H$ |
| 196 | ICPWGA39 | FFFF B188$_H$ | INTPWGA39 | PWGA39 interrupt | PWGA39 | Edge | 10C4$_H$ | √ | √ | √ | | | +310$_H$ |
| 197 | ICPWGA40 | FFFF B18A$_H$ | INTPWGA40 | PWGA40 interrupt | PWGA40 | Edge | 10C5$_H$ | √ | √ | √ | | | +314$_H$ |
| 198 | ICPWGA41 | FFFF B18C$_H$ | INTPWGA41 | PWGA41 interrupt | PWGA41 | Edge | 10C6$_H$ | √ | √ | √ | | | +318$_H$ |
| 199 | ICPWGA42 | FFFF B18E$_H$ | INTPWGA42 | PWGA42 interrupt | PWGA42 | Edge | 10C7$_H$ | √ | √ | √ | | | +31C$_H$ |
| 200 | ICPWGA43 | FFFF B190$_H$ | INTPWGA43 | PWGA43 interrupt | PWGA43 | Edge | 10C8$_H$ | √ | √ | √ | | | +320$_H$ |
| 201 | ICPWGA44 | FFFF B192$_H$ | INTPWGA44 | PWGA44 interrupt | PWGA44 | Edge | 10C9$_H$ | √ | √ | √ | | | +324$_H$ |
| 202 | ICPWGA45 | FFFF B194$_H$ | INTPWGA45 | PWGA45 interrupt | PWGA45 | Edge | 10CA$_H$ | √ | √ | √ | | | +328$_H$ |
| 203 | ICPWGA46 | FFFF B196$_H$ | INTPWGA46 | PWGA46 interrupt | PWGA46 | Edge | 10CB$_H$ | √ | √ | √ | | | +32C$_H$ |
| 204 | ICPWGA47 | FFFF B198$_H$ | INTPWGA47 | PWGA47 interrupt | PWGA47 | Edge | 10CC$_H$ | √ | √ | √ | | | +330$_H$ |
| 205 | ICP9 | FFFF B19A$_H$ | INTP9 | External interrupt | Port | Edge | 10CD$_H$ | — | √ | √ | | | +334$_H$ |
| 206 | ICP13 | FFFF B19C$_H$ | INTP13 | External interrupt | Port | Edge | 10CE$_H$ | √ | √ | √ | | | +338$_H$ |
| 207 | ICP14 | FFFF B19E$_H$ | INTP14 | External interrupt | Port | Edge | 10CF$_H$ | — | √ | √ | | | +33C$_H$ |
| 208 | ICP15 | FFFF B1A0$_H$ | INTP15 | External interrupt | Port | Edge | 10D0$_H$ | — | √ | √ | | | +340$_H$ |
| 209 | ICRTCA01S | FFFF B1A2$_H$ | INTRTCA01S | 1 second interrupt | RTCA0 | Edge | 10D1$_H$ | — | √ | √ | | | +344$_H$ |
| 210 | ICRTCA0AL | FFFF B1A4$_H$ | INTRTCA0AL | Alarm interrupt | RTCA0 | Edge | 10D2$_H$ | — | √ | √ | | | +348$_H$ |
| 211 | ICRTCA0R | FFFF B1A6$_H$ | INTRTCA0R | Periodic interrupt | RTCA0 | Edge | 10D3$_H$ | — | √ | √ | | | +34C$_H$ |
| 212 | ICADCA1ERR | FFFF B1A8$_H$ | INTADCA1ERR | ADCA1 error interrupt | ADCA1 | Edge | 10D4$_H$ | — | √ | √ | | | +350$_H$ |
| 213 | ICADCA1I0 | FFFF B1AA$_H$ | INTADCA1I0 | ADCA1 scan group 1 (SG1) end interrupt | ADCA1 | Edge | 10D5$_H$ | — | √ | √ | | | +354$_H$ |
| 214 | ICADCA1I1 | FFFF B1AC$_H$ | INTADCA1I1 | ADCA1 scan group 2 (SG2) end interrupt | ADCA1 | Edge | 10D6$_H$ | — | √ | √ | | | +358$_H$ |
| 215 | ICADCA1I2 | FFFF B1AE$_H$ | INTADCA1I2 | ADCA1 scan group 3 (SG3) end interrupt | ADCA1 | Edge | 10D7$_H$ | — | √ | √ | | | +35C$_H$ |
| 216 | Reserved | FFFF B1B0$_H$ | | | | | 10D8$_H$ | — | — | — | | | +360$_H$ |
| 217 | ICRCAN2ERR | FFFF B1B2$_H$ | INTRCAN2ERR | CAN2 error interrupt | RSCAN0 | Level | 10D9$_H$ | √ | √ | √ | | | +364$_H$ |
| 218 | ICRCAN2REC | FFFF B1B4$_H$ | INTRCAN2REC | CAN2 transmit/receive FIFO receive complete interrupt | RSCAN0 | Level | 10DA$_H$ | √ | √ | √ | | | +368$_H$ |
| 219 | ICRCAN2TRX | FFFF B1B6$_H$ | INTRCAN2TRX | CAN2 transmit interrupt | RSCAN0 | Level | 10DB$_H$ | √ | √ | √ | | | +36C$_H$ |
| 220 | ICRCAN3ERR | FFFF B1B8$_H$ | INTRCAN3ERR | CAN3 error interrupt | RSCAN0 | Level | 10DC$_H$ | √ | √ | √ | | | +370$_H$ |
| 221 | ICRCAN3REC | FFFF B1BA$_H$ | INTRCAN3REC | CAN3 transmit/receive FIFO receive complete interrupt | RSCAN0 | Level | 10DD$_H$ | √ | √ | √ | | | +374$_H$ |
| 222 | ICRCAN3TRX | FFFF B1BC$_H$ | INTRCAN3TRX | CAN3 transmit interrupt | RSCAN0 | Level | 10DE$_H$ | √ | √ | √ | | | +378$_H$ |
| 223 | ICCSIG1IC | FFFF B1BE$_H$ | INTCSIG1IC | CSIG1 communication status interrupt | CSIG1 | Edge | 10DF$_H$ | — | √ | √ | | | +37C$_H$ |
| 224 | ICCSIG1IR | FFFF B1C0$_H$ | INTCSIG1IR | CSIG1 receive status interrupt | CSIG1 | Edge | 10E0$_H$ | — | √ | √ | | | +380$_H$ |
| 225 | ICCSIG1IRE | FFFF B1C2$_H$ | INTCSIG1IRE | CSIG1 receive error interrupt | CSIG1 | Edge | 10E1$_H$ | — | √ | √ | | | +384$_H$ |
| 226 | ICRLIN24 | FFFF B1C4$_H$ | INTRLIN24 | RLIN24 interrupt | RLIN241 | Edge | 10E2$_H$ | — | √ | √ | | | +388$_H$ |
| 227 | ICRLIN25 | FFFF B1C6$_H$ | INTRLIN25 | RLIN25 interrupt | RLIN241 | Edge | 10E3$_H$ | — | √ | √ | | | +38C$_H$ |
| 228 | ICRLIN33 | FFFF B1C8$_H$ | INTRLIN33 | RLIN33 interrupt | RLIN33 | Edge | 10E4$_H$ | √ | √ | √ | | | +390$_H$ |
| 229 | ICRLIN33UR0 | FFFF B1CA$_H$ | INTRLIN33UR0 | RLIN33 transmit interrupt | RLIN33 | Edge | 10E5$_H$ | √ | √ | √ | | | +394$_H$ |
| 230 | ICRLIN33UR1 | FFFF B1CC$_H$ | INTRLIN33UR1 | RLIN33 receive complete interrupt | RLIN33 | Edge | 10E6$_H$ | √ | √ | √ | | | +398$_H$ |
| 231 | ICRLIN33UR2 | FFFF B1CE$_H$ | INTRLIN33UR2 | RLIN33 status interrupt | RLIN33 | Edge | 10E7$_H$ | √ | √ | √ | | | +39C$_H$ |
| 232 | ICRLIN34 | FFFF B1D0$_H$ | INTRLIN34 | RLIN34 interrupt | RLIN34 | Edge | 10E8$_H$ | — | √ | √ | | | +3A0$_H$ |

Table 7.4 EI Level Maskable Interrupt Sources (7/9)

| Interrupt | | | Interrupt Request | | | | Exception Source Code | 100 pins | 144 pins | 176 pins | Handler Address (Offset)*8 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Channel No.*1 | Control Register | | Name | Source | Unit | Detection Type*2 | | | | | Direct Jumping to an Address | | Reference to a Table*5 |
| | Name | Address | | | | | | | | | RINT = 0 | RINT = 1 | |
| 233 | ICRLIN34UR0 | FFFF B1D2$_H$ | INTRLIN34UR0 | RLIN34 transmit interrupt | RLIN34 | Edge | 10E9$_H$ | — | √ | √ | *3 | *4 | +3A4$_H$ |
| 234 | ICRLIN34UR1 | FFFF B1D4$_H$ | INTRLIN34UR1 | RLIN34 receive complete interrupt | RLIN34 | Edge | 10EA$_H$ | — | √ | √ | | | +3A8$_H$ |
| 235 | ICRLIN34UR2 | FFFF B1D6$_H$ | INTRLIN34UR2 | RLIN34 status interrupt | RLIN34 | Edge | 10EB$_H$ | — | √ | √ | | | +3AC$_H$ |
| 236 | ICRLIN35 | FFFF B1D8$_H$ | INTRLIN35 | RLIN35 interrupt | RLIN35 | Edge | 10EC$_H$ | — | √ | √ | | | +3B0$_H$ |
| 237 | ICRLIN35UR0 | FFFF B1DA$_H$ | INTRLIN35UR0 | RLIN35 transmit interrupt | RLIN35 | Edge | 10ED$_H$ | — | √ | √ | | | +3B4$_H$ |
| 238 | ICRLIN35UR1 | FFFF B1DC$_H$ | INTRLIN35UR1 | RLIN35 receive complete interrupt | RLIN35 | Edge | 10EE$_H$ | — | √ | √ | | | +3B8$_H$ |
| 239 | ICRLIN35UR2 | FFFF B1DE$_H$ | INTRLIN35UR2 | RLIN35 status interrupt | RLIN35 | Edge | 10EF$_H$ | — | √ | √ | | | +3BC$_H$ |
| 240 | ICPWGA48 | FFFF B1E0$_H$ | INTPWGA48 | PWGA48 interrupt | PWGA48 | Edge | 10F0$_H$ | — | √ | √ | | | +3C0$_H$ |
| 241 | ICPWGA49 | FFFF B1E2$_H$ | INTPWGA49 | PWGA49 interrupt | PWGA49 | Edge | 10F1$_H$ | — | √ | √ | | | +3C4$_H$ |
| 242 | ICPWGA50 | FFFF B1E4$_H$ | INTPWGA50 | PWGA50 interrupt | PWGA50 | Edge | 10F2$_H$ | — | √ | √ | | | +3C8$_H$ |
| 243 | ICPWGA51 | FFFF B1E6$_H$ | INTPWGA51 | PWGA51 interrupt | PWGA51 | Edge | 10F3$_H$ | — | √ | √ | | | +3CC$_H$ |
| 244 | ICPWGA52 | FFFF B1E8$_H$ | INTPWGA52 | PWGA52 interrupt | PWGA52 | Edge | 10F4$_H$ | — | √ | √ | | | +3D0$_H$ |
| 245 | ICPWGA53 | FFFF B1EA$_H$ | INTPWGA53 | PWGA53 interrupt | PWGA53 | Edge | 10F5$_H$ | — | √ | √ | | | +3D4$_H$ |
| 246 | ICPWGA54 | FFFF B1EC$_H$ | INTPWGA54 | PWGA54 interrupt | PWGA54 | Edge | 10F6$_H$ | — | √ | √ | | | +3D8$_H$ |
| 247 | ICPWGA55 | FFFF B1EE$_H$ | INTPWGA55 | PWGA55 interrupt | PWGA55 | Edge | 10F7$_H$ | — | √ | √ | | | +3DC$_H$ |
| 248 | ICPWGA56 | FFFF B1F0$_H$ | INTPWGA56 | PWGA56 interrupt | PWGA56 | Edge | 10F8$_H$ | — | √ | √ | | | +3E0$_H$ |
| 249 | ICPWGA57 | FFFF B1F2$_H$ | INTPWGA57 | PWGA57 interrupt | PWGA57 | Edge | 10F9$_H$ | — | √ | √ | | | +3E4$_H$ |
| 250 | ICPWGA58 | FFFF B1F4$_H$ | INTPWGA58 | PWGA58 interrupt | PWGA58 | Edge | 10FA$_H$ | — | √ | √ | | | +3E8$_H$ |
| 251 | ICPWGA59 | FFFF B1F6$_H$ | INTPWGA59 | PWGA59 interrupt | PWGA59 | Edge | 10FB$_H$ | — | √ | √ | | | +3EC$_H$ |
| 252 | ICPWGA60 | FFFF B1F8$_H$ | INTPWGA60 | PWGA60 interrupt | PWGA60 | Edge | 10FC$_H$ | — | √ | √ | | | +3F0$_H$ |
| 253 | ICPWGA61 | FFFF B1FA$_H$ | INTPWGA61 | PWGA61 interrupt | PWGA61 | Edge | 10FD$_H$ | — | √ | √ | | | +3F4$_H$ |
| 254 | ICPWGA62 | FFFF B1FC$_H$ | INTPWGA62 | PWGA62 interrupt | PWGA62 | Edge | 10FE$_H$ | — | √ | √ | | | +3F8$_H$ |
| 255 | ICPWGA63 | FFFF B1FE$_H$ | INTPWGA63 | PWGA63 interrupt | PWGA63 | Edge | 10FF$_H$ | — | √ | √ | | | +3FC$_H$ |
| 256 | ICTAUB1I0 | FFFF B200$_H$ | INTTAUB1I0 | Interrupt for TAUB1 channel 0 | TAUB1 | Edge | 1100$_H$ | — | — | √ | | | +400$_H$ |
| 257 | ICTAUB1I1 | FFFF B202$_H$ | INTTAUB1I1 | Interrupt for TAUB1 channel 1 | TAUB1 | Edge | 1101$_H$ | — | — | √ | | | +404$_H$ |
| 258 | ICTAUB1I2 | FFFF B204$_H$ | INTTAUB1I2 | Interrupt for TAUB1 channel 2 | TAUB1 | Edge | 1102$_H$ | — | — | √ | | | +408$_H$ |
| 259 | ICTAUB1I3 | FFFF B206$_H$ | INTTAUB1I3 | Interrupt for TAUB1 channel 3 | TAUB1 | Edge | 1103$_H$ | — | — | √ | | | +40C$_H$ |
| 260 | ICTAUB1I4 | FFFF B208$_H$ | INTTAUB1I4 | Interrupt for TAUB1 channel 4 | TAUB1 | Edge | 1104$_H$ | — | — | √ | | | +410$_H$ |
| 261 | ICTAUB1I5 | FFFF B20A$_H$ | INTTAUB1I5 | Interrupt for TAUB1 channel 5 | TAUB1 | Edge | 1105$_H$ | — | — | √ | | | +414$_H$ |
| 262 | ICTAUB1I6 | FFFF B20C$_H$ | INTTAUB1I6 | Interrupt for TAUB1 channel 6 | TAUB1 | Edge | 1106$_H$ | — | — | √ | | | +418$_H$ |
| 263 | ICTAUB1I7 | FFFF B20E$_H$ | INTTAUB1I7 | Interrupt for TAUB1 channel 7 | TAUB1 | Edge | 1107$_H$ | — | — | √ | | | +41C$_H$ |
| 264 | ICTAUB1I8 | FFFF B210$_H$ | INTTAUB1I8 | Interrupt for TAUB1 channel 8 | TAUB1 | Edge | 1108$_H$ | — | — | √ | | | +420$_H$ |
| 265 | ICTAUB1I9 | FFFF B212$_H$ | INTTAUB1I9 | Interrupt for TAUB1 channel 9 | TAUB1 | Edge | 1109$_H$ | — | — | √ | | | +424$_H$ |
| 266 | ICTAUB1I10 | FFFF B214$_H$ | INTTAUB1I10 | Interrupt for TAUB1 channel 10 | TAUB1 | Edge | 110A$_H$ | — | — | √ | | | +428$_H$ |
| 267 | ICTAUB1I11 | FFFF B216$_H$ | INTTAUB1I11 | Interrupt for TAUB1 channel 11 | TAUB1 | Edge | 110B$_H$ | — | — | √ | | | +42C$_H$ |
| 268 | ICTAUB1I12 | FFFF B218$_H$ | INTTAUB1I12 | Interrupt for TAUB1 channel 12 | TAUB1 | Edge | 110C$_H$ | — | — | √ | | | +430$_H$ |
| 269 | ICTAUB1I13 | FFFF B21A$_H$ | INTTAUB1I13 | Interrupt for TAUB1 channel 13 | TAUB1 | Edge | 110D$_H$ | — | — | √ | | | +434$_H$ |
| 270 | ICTAUB1I14 | FFFF B21C$_H$ | INTTAUB1I14 | Interrupt for TAUB1 channel 14 | TAUB1 | Edge | 110E$_H$ | — | — | √ | | | +438$_H$ |
| 271 | ICTAUB1I15 | FFFF B21E$_H$ | INTTAUB1I15 | Interrupt for TAUB1 channel 15 | TAUB1 | Edge | 110F$_H$ | — | — | √ | | | +43C$_H$ |
| 272 | ICRCAN4ERR | FFFF B220$_H$ | INTRCAN4ERR | CAN4 error interrupt | RSCAN0 | Level | 1110$_H$ | √ | √ | √ | | | +440$_H$ |
| 273 | ICRCAN4REC | FFFF B222$_H$ | INTRCAN4REC | CAN4 transmit/receive FIFO receive complete interrupt | RSCAN0 | Level | 1111$_H$ | √ | √ | √ | | | +444$_H$ |
| 274 | ICRCAN4TRX | FFFF B224$_H$ | INTRCAN4TRX | CAN4 transmit interrupt | RSCAN0 | Level | 1112$_H$ | √ | √ | √ | | | +448$_H$ |
| 275 | ICRLIN26 | FFFF B226$_H$ | INTRLIN26 | RLIN26 interrupt | RLIN241 | Edge | 1113$_H$ | — | — | √ | | | +44C$_H$ |
| 276 | ICRLIN27 | FFFF B228$_H$ | INTRLIN27 | RLIN27 interrupt | RLIN241 | Edge | 1114$_H$ | — | — | √ | | | +450$_H$ |
| 277 | ICPWGA64 | FFFF B22A$_H$ | INTPWGA64 | PWGA64 interrupt | PWGA64 | Edge | 1115$_H$ | — | — | √ | | | +454$_H$ |
| 278 | ICPWGA65 | FFFF B22C$_H$ | INTPWGA65 | PWGA65 interrupt | PWGA65 | Edge | 1116$_H$ | — | — | √ | | | +458$_H$ |

Table 7.4 EI Level Maskable Interrupt Sources (8/9)

| Interrupt | | | Interrupt Request | | | | Exception Source Code | 100 pins | 144 pins | 176 pins | Handler Address (Offset)*8 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Channel No.*1 | Control Register | | Name | Source | Unit | Detection Type*2 | | | | | Direct Jumping to an Address | | Reference to a Table*5 |
| | Name | Address | | | | | | | | | RINT = 0 | RINT = 1 | |
| 279 | ICPWGA66 | FFFF B22E$_H$ | INTPWGA66 | PWGA66 interrupt | PWGA66 | Edge | 1117$_H$ | — | — | √ | *3 | *4 | +45C$_H$ |
| 280 | ICPWGA67 | FFFF B230$_H$ | INTPWGA67 | PWGA67 interrupt | PWGA67 | Edge | 1118$_H$ | — | — | √ | | | +460$_H$ |
| 281 | ICPWGA68 | FFFF B232$_H$ | INTPWGA68 | PWGA68 interrupt | PWGA68 | Edge | 1119$_H$ | — | — | √ | | | +464$_H$ |
| 282 | ICPWGA69 | FFFF B234$_H$ | INTPWGA69 | PWGA69 interrupt | PWGA69 | Edge | 111A$_H$ | — | — | √ | | | +468$_H$ |
| 283 | ICPWGA70 | FFFF B236$_H$ | INTPWGA70 | PWGA70 interrupt | PWGA70 | Edge | 111B$_H$ | — | — | √ | | | +46C$_H$ |
| 284 | ICPWGA71 | FFFF B238$_H$ | INTPWGA71 | PWGA71 interrupt | PWGA71 | Edge | 111C$_H$ | — | — | √ | | | +470$_H$ |
| 285 | ICRLIN28 | FFFF B23A$_H$ | INTRLIN28 | RLIN28 interrupt | RLIN242 | Edge | 111D$_H$ | — | — | √ | | | +474$_H$ |
| 286 | ICRLIN29 | FFFF B23C$_H$ | INTRLIN29 | RLIN29 interrupt | RLIN242 | Edge | 111E$_H$ | — | — | √ | | | +478$_H$ |
| 287 | ICRCAN5ERR | FFFF B23E$_H$ | INTRCAN5ERR | CAN5 error interrupt | RSCAN0 | Level | 111F$_H$ | √ | √ | √ | | | +47C$_H$ |
| 288 | ICRCAN5REC | FFFF B240$_H$ | INTRCAN5REC | CAN5 transmit/receive FIFO receive complete interrupt | RSCAN0 | Level | 1120$_H$ | √ | √ | √ | | | +480$_H$ |
| 289 | ICRCAN5TRX | FFFF B242$_H$ | INTRCAN5TRX | CAN5 transmit interrupt | RSCAN0 | Level | 1121$_H$ | √ | √ | √ | | | +484$_H$ |
| 290 | Reserved | FFFF B244$_H$ | | | | | 1122$_H$ | — | — | — | | | +488$_H$ |
| 291 | Reserved | FFFF B246$_H$ | | | | | 1123$_H$ | — | — | — | | | +48C$_H$ |
| 292 | Reserved | FFFF B248$_H$ | | | | | 1124$_H$ | — | — | — | | | +490$_H$ |
| 293 | Reserved | FFFF B24A$_H$ | | | | | 1125$_H$ | — | — | — | | | +494$_H$ |
| 294 | Reserved | FFFF B24C$_H$ | | | | | 1126$_H$ | — | — | — | | | +498$_H$ |
| 295 | Reserved | FFFF B24E$_H$ | | | | | 1127$_H$ | — | — | — | | | +49C$_H$ |
| 296 | Reserved | FFFF B250$_H$ | | | | | 1128$_H$ | — | — | — | | | +4A0$_H$ |
| 297 | Reserved | FFFF B252$_H$ | | | | | 1129$_H$ | — | — | — | | | +4A4$_H$ |
| 298 | Reserved | FFFF B254$_H$ | | | | | 112A$_H$ | — | — | — | | | +4A8$_H$ |
| 299 | Reserved | FFFF B256$_H$ | | | | | 112B$_H$ | — | — | — | | | +4AC$_H$ |
| 300 | Reserved | FFFF B258$_H$ | | | | | 112C$_H$ | — | — | — | | | +4B0$_H$ |
| 301 | Reserved | FFFF B25A$_H$ | | | | | 112D$_H$ | — | — | — | | | +4B4$_H$ |
| 302 | Reserved | FFFF B25C$_H$ | | | | | 112E$_H$ | — | — | — | | | +4B8$_H$ |
| 303 | Reserved | FFFF B25E$_H$ | | | | | 112F$_H$ | — | — | — | | | +4BC$_H$ |
| 304 | Reserved | FFFF B260$_H$ | | | | | 1130$_H$ | — | — | — | | | +4C0$_H$ |
| 305 | Reserved | FFFF B262$_H$ | | | | | 1131$_H$ | — | — | — | | | +4C4$_H$ |
| 306 | Reserved | FFFF B264$_H$ | | | | | 1132$_H$ | — | — | — | | | +4C8$_H$ |
| 307 | Reserved | FFFF B266$_H$ | | | | | 1133$_H$ | — | — | — | | | +4CC$_H$ |
| 308 | Reserved | FFFF B268$_H$ | | | | | 1134$_H$ | — | — | — | | | +4D0$_H$ |
| 309 | Reserved | FFFF B26A$_H$ | | | | | 1135$_H$ | — | — | — | | | +4D4$_H$ |
| 310 | Reserved | FFFF B26C$_H$ | | | | | 1136$_H$ | — | — | — | | | +4D8$_H$ |
| 311 | Reserved | FFFF B26E$_H$ | | | | | 1137$_H$ | — | — | — | | | +4DC$_H$ |
| 312 | Reserved | FFFF B270$_H$ | | | | | 1138$_H$ | — | — | — | | | +4E0$_H$ |
| 313 | Reserved | FFFF B272$_H$ | | | | | 1139$_H$ | — | — | — | | | +4E4$_H$ |
| 314 | Reserved | FFFF B274$_H$ | | | | | 113A$_H$ | — | — | — | | | +4E8$_H$ |
| 315 | Reserved | FFFF B276$_H$ | | | | | 113B$_H$ | — | — | — | | | +4EC$_H$ |
| 316 | Reserved | FFFF B278$_H$ | | | | | 113C$_H$ | — | — | — | | | +4F0$_H$ |
| 317 | Reserved | FFFF B27A$_H$ | | | | | 113D$_H$ | — | — | — | | | +4F4$_H$ |
| 318 | Reserved | FFFF B27C$_H$ | | | | | 113E$_H$ | — | — | — | | | +4F8$_H$ |
| 319 | ICRCANGERR1 | FFFF B27E$_H$ | INTRCANGERR1 | CAN global error interrupt | RSCAN1 | Level | 113F$_H$ | — | — | √ | | | +4FC$_H$ |
| 320 | ICRCANGRECC1 | FFFF B280$_H$ | INTRCANGRECC1 | CAN receive FIFO interrupt | RSCAN1 | Level | 1140$_H$ | — | — | √ | | | +500$_H$ |
| 321 | ICRCAN6ERR | FFFF B282$_H$ | INTRCAN6ERR | CAN6 error interrupt | RSCAN1 | Level | 1141$_H$ | — | — | √ | | | +504$_H$ |
| 322 | ICRCAN6REC | FFFF B284$_H$ | INTRCAN6REC | CAN6 transmit/receive FIFO receive complete interrupt | RSCAN1 | Level | 1142$_H$ | — | — | √ | | | +508$_H$ |
| 323 | ICRCAN6TRX | FFFF B286$_H$ | INTRCAN6TRX | CAN6 transmit interrupt | RSCAN1 | Level | 1143$_H$ | — | — | √ | | | +50C$_H$ |

**Table 7.4 EI Level Maskable Interrupt Sources (9/9)**

| Interrupt | | | Interrupt Request | | | | Exception Source Code | 100 pins | 144 pins | 176 pins | Handler Address (Offset)*8 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Control Register | | | | | | | | | | Direct Jumping to an Address | | |
| Channel No.*1 | Name | Address | Name | Source | Unit | Detection Type*2 | | | | | RINT = 0 | RINT = 1 | Reference to a Table*5 |
| 324 | Reserved | FFFF B288$_H$ | | | | | 1144$_H$ | — | — | — | *3 | *4 | +510$_H$ |
| 325 | Reserved | FFFF B28A$_H$ | | | | | 1145$_H$ | — | — | — | | | +514$_H$ |
| 326 | Reserved | FFFF B28C$_H$ | | | | | 1146$_H$ | — | — | — | | | +518$_H$ |
| 327 | Reserved | FFFF B28E$_H$ | | | | | 1147$_H$ | — | — | — | | | +51C$_H$ |
| 328 | Reserved | FFFF B290$_H$ | | | | | 1148$_H$ | — | — | — | | | +520$_H$ |
| 329 | Reserved | FFFF B292$_H$ | | | | | 1149$_H$ | — | — | — | | | +524$_H$ |
| 330 | Reserved | FFFF B294$_H$ | | | | | 114A$_H$ | — | — | — | | | +528$_H$ |
| 331 | Reserved | FFFF B296$_H$ | | | | | 114B$_H$ | — | — | — | | | +52C$_H$ |
| 332 | Reserved | FFFF B298$_H$ | | | | | 114C$_H$ | — | — | — | | | +530$_H$ |
| 333 | Reserved | FFFF B29A$_H$ | | | | | 114D$_H$ | — | — | — | | | +534$_H$ |
| 334 | Reserved | FFFF B29C$_H$ | | | | | 114E$_H$ | — | — | — | | | +538$_H$ |
| 335 | Reserved | FFFF B29E$_H$ | | | | | 114F$_H$ | — | — | — | | | +53C$_H$ |
| 336 | Reserved | FFFF B2A0$_H$ | | | | | 1150$_H$ | — | — | — | | | +540$_H$ |
| 337 | Reserved | FFFF B2A2$_H$ | | | | | 1151$_H$ | — | — | — | | | +544$_H$ |
| 338 | Reserved | FFFF B2A4$_H$ | | | | | 1152$_H$ | — | — | — | | | +548$_H$ |
| 339 | Reserved | FFFF B2A6$_H$ | | | | | 1153$_H$ | — | — | — | | | +54C$_H$ |
| 340 | Reserved | FFFF B2A8$_H$ | | | | | 1154$_H$ | — | — | — | | | +550$_H$ |
| 341 | Reserved | FFFF B2AA$_H$ | | | | | 1155$_H$ | — | — | — | | | +554$_H$ |
| 342 | Reserved | FFFF B2AC$_H$ | | | | | 1156$_H$ | — | — | — | | | +558$_H$ |
| 343 | Reserved | FFFF B2AE$_H$ | | | | | 1157$_H$ | — | — | — | | | +55C$_H$ |
| 344 | Reserved | FFFF B2B0$_H$ | | | | | 1158$_H$ | — | — | — | | | +560$_H$ |
| 345 | Reserved | FFFF B2B2$_H$ | | | | | 1159$_H$ | — | — | — | | | +564$_H$ |
| 346 | Reserved | FFFF B2B4$_H$ | | | | | 115A$_H$ | — | — | — | | | +568$_H$ |
| 347 | Reserved | FFFF B2B6$_H$ | | | | | 115B$_H$ | — | — | — | | | +56C$_H$ |
| 348 | Reserved | FFFF B2B8$_H$ | | | | | 115C$_H$ | — | — | — | | | +570$_H$ |
| 349 | Reserved | FFFF B2BA$_H$ | | | | | 115D$_H$ | — | — | — | | | +574$_H$ |
| 350 | Reserved | FFFF B2BC$_H$ | | | | | 115E$_H$ | — | — | — | | | +578$_H$ |
| 351 | Reserved | FFFF B2BE$_H$ | | | | | 115F$_H$ | — | — | — | | | +57C$_H$ |
| 352 | Reserved | FFFF B2C0$_H$ | | | | | 1160$_H$ | — | — | — | | | +580$_H$ |
| 353 | Reserved | FFFF B2C2$_H$ | | | | | 1161$_H$ | — | — | — | | | +584$_H$ |
| 354 | Reserved | FFFF B2C4$_H$ | | | | | 1162$_H$ | — | — | — | | | +588$_H$ |
| 355 | Reserved | FFFF B2C6$_H$ | | | | | 1163$_H$ | — | — | — | | | +58C$_H$ |
| 356 | ICDPE | FFFF B2C8$_H$ | INTDPE | LPS digital port error interrupt | LPS0 | Level | 1164$_H$ | √ | √ | √ | | | +590$_H$ |
| 357 | ICAPE | FFFF B2CA$_H$ | INTAPE | LPS analog port error interrupt | LPS0 | Level | 1165$_H$ | √ | √ | √ | | | +594$_H$ |

Note 1. Each interrupt is connected to INTC1 channel 8 to 31 and INTC2 channel 32 to 357.

Note 2. This indicates whether an interrupt source is detected at the level or edge. This also affects the value after reset of an EI level interrupt control register. For details, see **Section 7.4.2, ICxxx — EI Level Interrupt Control Registers**. For detection at level, an interrupt source is cleared by accessing to the register that retains an interrupt source. Dummy-reading of the register and execution of the SYNCP instruction are required to reflect the result of the register update to the subsequent instruction.

Note 3. Irrespective of interrupt channels, an offset address is determined in the range from +100$_H$ to 1F0$_H$ according to the priority (0 to 15).

Note 4. Irrespective of the priority, offset addresses are uniformly +100$_H$.

Note 5. The table reference method uses a table for reading an exception handler address for each interrupt channel, and it extracts handler address by referencing that table. Table reference position is determined by the following formula.
Exception handler address read position = INTBP register + channel number × 4 bytes

Note 6. The same interrupt source is assigned to different interrupt channels. For details, see **Section 7.5.2, SELB_INTC1 — INTC1 Interrupt Select Register**.

Note 7. Two interrupt sources are assigned to the same interrupt channel. For details, see **Section 7.5.3, SELB_INTC2 — INTC2 Interrupt Select Register**.

Note 8. For details, see **Section 7.10, Exception Handler Address**.
Note 9. INTOSTM0 can operate as an EIINT or FEINT interrupt, but using it in both ways at the same time is not possible. It is used as FEINT when OSTM0 functions as the TSU (timing supervision unit). It is used as EIINT when OSTM0 functions as anything other than the TSU.
Note 10. For details on the interrupt source, see the *RH850/F1K Flash Memory User's Manual: Hardware Interface*.

### 7.2.2 List of Registers

Interrupt sources registers are listed in the following table.

**Table 7.5 List of Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| ECON_NMI | FENMI Factor Register | WDTNMIF | FFC0 $0000_H$ |
| | WDTNMI Factor Clear Register | WDTNMIFC | FFC0 $0008_H$ |
| ECON_FEINT | FEINT Factor Register | FEINTF | FFC0 $0100_H$ |
| | FEINT Factor Mask Register | FEINTFMSK | FFC0 $0104_H$ |
| | FEINT Factor Clear Register | FEINTFC | FFC0 $0108_H$ |

## 7.2.3 FE Level Non-Maskable Interrupt Sources

### 7.2.3.1 WDTNMIF — FENMI Factor Register

This register contains information about which source has generated the FE level non-maskable interrupt (FENMI). This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFC0 0000$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | WDTA1 NMIF | WDTA0 NMIF | TNMIF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.6 WDTNMIF Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 | WDTA1NMIF | WDTA1NMI flag<br>0: No WDTA1NMI occurred<br>1: WDTA1NMI has occurred |
| 1 | WDTA0NMIF | WDTA0NMI flag<br>0: No WDTA0NMI occurred<br>1: WDTA0NMI has occurred |
| 0 | TNMIF | Input signal flag from the NMI pin<br>0: No TNMI occurred<br>1: TNMI has occurred |

### 7.2.3.2 WDTNMIFC — WDTNMI Factor Clear Register

This register clears the FE level non-maskable interrupt flags of the WDTNMIF register.

**Access:** This register is a write-only register that can be written in 32-bit units.

**Address:** FFC0 0008$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | WDTA1NMIFC | WDTA0NMIFC | TNMIFC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | W | W | W |

**Table 7.7 WDTNMIFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When writing, write the value after reset. |
| 2 | WDTA1NMIFC | WDTA1NMIF flag clear<br>0: —<br>1: Clear |
| 1 | WDTA0NMIFC | WDTA0NMIF flag clear<br>0: —<br>1: Clear |
| 0 | TNMIFC | TNMIF flag clear<br>0: —<br>1: Clear |

## 7.2.4 FE Level Maskable Interrupt Sources

### 7.2.4.1 FEINTF — FEINT Factor Register

This register contains information about which source has generated the FE level maskable interrupt (FEINT). This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFC0 0100$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMAFEIF | — | — | — | — | — | — | — | — | ECCDCNRAM1FEIF | OSTM4FEIF | OSTM3FEIF | OSTM2FEIF | OSTM1FEIF | GUARDFEIF | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LVIHFEIF | OSTM0FEIF | ECCRAMFEIF | ECCSCFLI0FEIF | — | ECCDCSIH3FEIF | ECCDCSIH2FEIF | ECCDCSIH1FEIF | ECCDCSIH0FEIF | ECCDCNRAM0FEIF | — | ECCDEEP0FEIF | — | — | — | LVILFEIF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.8 FEINTF Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | DMAFEIF | INTDMAERR interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 30 to 23 | Reserved | When read, the value after reset is returned. |
| 22 | ECCDCNRAM1FEIF*1 | INTECCDCNRAM1 interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 21 | OSTM4FEIF | INTOSTM4_FE interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 20 | OSTM3FEIF | INTOSTM3_FE interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 19 | OSTM2FEIF | INTOSTM2_FE interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 18 | OSTM1FEIF | INTOSTM1_FE interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 17 | GUARDFEIF | INTGUARD interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 16 | Reserved | When read, the value after reset is returned. |
| 15 | LVIHFEIF | INTLVIH interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 14 | OSTM0FEIF | INTOSTM0_FE interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |

Table 7.8 FEINTF Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 13 | ECCRAM FEIF | INTECCRAM interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 12 | ECCSCFLI0 FEIF | INTECCSCFLI0 interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 11 | Reserved | When read, the value after reset is returned. |
| 10 | ECCDCSIH3 FEIF | INTECCDCSIH3 interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 9 | ECCDCSIH2 FEIF | INTECCDCSIH2 interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 8 | ECCDCSIH1 FEIF | INTECCDCSIH1 interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 7 | ECCDCSIH0 FEIF | INTECCDCSIH0 interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 6 | ECCDCNRAM0 FEIF | INTECCDCNRAM0 interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 5 | Reserved | When read, the value after reset is returned. |
| 4 | ECCDEEP0 FEIF | INTECCDEEP0 interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 3 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | LVILFEIF | INTLVIL interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |

Note 1. For the supported products, see **Table 7.3, FE Level Maskable Interrupt Requests**.

### 7.2.4.2 FEINTFMSK — FEINT Factor Mask Register

This register masks the FE level maskable interrupt (FEINT). This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFC0 0104$_H$

**Value after reset:** FFFF FFFF$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMAFEIFMSK | — | — | — | — | — | — | — | — | ECCDCNRAM1FEIFMSK | OSTM4FEIFMSK | OSTM3FEIFMSK | OSTM2FEIFMSK | OSTM1FEIFMSK | GUARDFEIFMSK | — | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LVIHFEIFMSK | OSTM0FEIFMSK | ECCRAMFEIFMSK | ECCSCFLI0FEIFMSK | — | ECCDCSIH3FEIFMSK | ECCDCSIH2FEIFMSK | ECCDCSIH1FEIFMSK | ECCDCSIH0FEIFMSK | ECCDCNRAM0FEIFMSK | — | ECCDEEP0FEIFMSK | — | — | — | LVILFEIFMSK |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R | R/W | R | R | R | R/W |

**Table 7.9 FEINTFMSK Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | DMAFEIFMSK | INTDMAERR interrupt mask<br>0: Not masked<br>1: Masked |
| 30 to 23 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 22 | ECCDCNRAM1FEIFMSK*1 | INTECCDCNRAM1 interrupt mask<br>0: Not masked<br>1: Masked |
| 21 | OSTM4FEIFMSK | INTOSTM4_FE interrupt mask<br>0: Not masked<br>1: Masked |
| 20 | OSTM3FEIFMSK | INTOSTM3_FE interrupt mask<br>0: Not masked<br>1: Masked |
| 19 | OSTM2FEIFMSK | INTOSTM2_FE interrupt mask<br>0: Not masked<br>1: Masked |
| 18 | OSTM1FEIFMSK | INTOSTM1_FE interrupt mask<br>0: Not masked<br>1: Masked |
| 17 | GUARDFEIFMSK | INTGUARD interrupt mask<br>0: Not masked<br>1: Masked |
| 16 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 15 | LVIHFEIFMSK | INTLVIH interrupt mask<br>0: Not masked<br>1: Masked |
| 14 | OSTM0FEIFMSK | INTOSTM0_FE interrupt mask<br>0: Not masked<br>1: Masked |
| 13 | ECCRAMFEIFMSK | INTECCRAM interrupt mask<br>0: Not masked<br>1: Masked |

Table 7.9 FEINTFMSK Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 12 | ECCSCFLI0 FEIFMSK | INTECCSCFLI0 interrupt mask<br>0: Not masked<br>1: Masked |
| 11 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 10 | ECCDCSIH3FEI FMSK | INTECCDCSIH3 interrupt mask<br>0: Not masked<br>1: Masked |
| 9 | ECCDCSIH2FEI FMSK | INTECCDCSIH2 interrupt mask<br>0: Not masked<br>1: Masked |
| 8 | ECCDCSIH1FEI FMSK | INTECCDCSIH1 interrupt mask<br>0: Not masked<br>1: Masked |
| 7 | ECCDCSIH0FEI FMSK | INTECCDCSIH0 interrupt mask<br>0: Not masked<br>1: Masked |
| 6 | ECCDCNRAM0 FEIFMSK | INTECCDCNRAM0 interrupt mask<br>0: Not masked<br>1: Masked |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 | ECCDEEP0FEI FMSK | INTECCDEEP0 interrupt mask<br>0: Not masked<br>1: Masked |
| 3 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | LVILFEIFMSK | INTLVIL interrupt mask<br>0: Not masked<br>1: Masked |

Note 1. For the supported products, see **Table 7.3, FE Level Maskable Interrupt Requests**.

### 7.2.4.3 FEINTFC — FEINT Factor Clear Register

This register clears the bits of the FEINT factor register (FEINTF).

**Access:** This register is a write-only register that can be written in 32-bit units.

**Address:** FFC0 0108$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMAFEIFC | — | — | — | — | — | — | — | — | ECCDCNRAM1FEIFC | OSTM4FEIFC | OSTM3FEIFC | OSTM2FEIFC | OSTM1FEIFC | GUARDFEIFC | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LVIHFEIFC | OSTM0FEIFC | ECCRAMFEIFC | ECCSCFLI0FEIFC | — | ECCDCSIH3FEIFC | ECCDCSIH2FEIFC | ECCDCSIH1FEIFC | ECCDCSIH0FEIFC | ECCDCNRAM0FEIFC | — | ECCDEEP0FEIFC | — | — | — | LVILFEIFC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | R | W | W | W | W | W | R | W | R | R | R | W |

**Table 7.10 FEINTFC Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | DMAFEIFC | DMAFEIF flag clear<br>0: —<br>1: Clear |
| 30 to 23 | Reserved | When writing, write the value after reset. |
| 22 | ECCDCNRAM1FEIFC*1 | ECCDCNRAM1FEIF flag clear<br>0: —<br>1: Clear |
| 21 | OSTM4FEIFC | OSTM4FEIF flag clear<br>0: —<br>1: Clear |
| 20 | OSTM3FEIFC | OSTM3FEIF flag clear<br>0: —<br>1: Clear |
| 19 | OSTM2FEIFC | OSTM2FEIF flag clear<br>0: —<br>1: Clear |
| 18 | OSTM1FEIFC | OSTM1FEIF flag clear<br>0: —<br>1: Clear |
| 17 | GUARDFEIFC | GUARDFEIF flag clear<br>0: —<br>1: Clear |
| 16 | Reserved | When writing, write the value after reset. |
| 15 | LVIHFEIFC | LVIHFEIF flag clear<br>0: —<br>1: Clear |
| 14 | OSTM0FEIFC | OSTM0FEIF flag clear<br>0: —<br>1: Clear |
| 13 | ECCRAMFEIFC | ECCRAMFEIF flag clear<br>0: —<br>1: Clear |
| 12 | ECCSCFLI0FEIFC | ECCSCFLI0FEIF flag clear<br>0: —<br>1: Clear |

Table 7.10 FEINTFC Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 | Reserved | When writing, write the value after reset. |
| 10 | ECCDCSIH3 FEIFC | ECCDCSIH3FEIF flag clear<br>0: —<br>1: Clear |
| 9 | ECCDCSIH2 FEIFC | ECCDCSIH2FEIF flag clear<br>0: —<br>1: Clear |
| 8 | ECCDCSIH1 FEIFC | ECCDCSIH1FEIF flag clear<br>0: —<br>1: Clear |
| 7 | ECCDCSIH0 FEIFC | ECCDCSIH0FEIF flag clear<br>0: —<br>1: Clear |
| 6 | ECCDCNRAM0 FEIFC | ECCDCNRAM0FEIF flag clear<br>0: —<br>1: Clear |
| 5 | Reserved | When writing, write the value after reset. |
| 4 | ECCDEEP0 FEIFC | ECCDEEP0FEIF flag clear<br>0: —<br>1: Clear |
| 3 to 1 | Reserved | When writing, write the value after reset. |
| 0 | LVILFEIFC | LVILFEIF flag clear<br>0: —<br>1: Clear |

Note 1. For the supported products, see **Table 7.3, FE Level Maskable Interrupt Requests**.

## 7.3 Edge/Level Detection

External interrupts (TNMI and INTPm) can be specified to be generated when a rising edge, falling edge, rising or falling edge, low level, or high level is detected at an external interrupt pin.

The following registers are used to specify the edge and level of each interrupt:

**Table 7.11 External Interrupt Edge/Level Detection Registers**

| Interrupt | Register |
|---|---|
| TNMI | FCLA0CTL0_NMI |
| INTP0 | FCLA0CTL0_INTPL |
| INTP1 | FCLA0CTL1_INTPL |
| INTP2 | FCLA0CTL2_INTPL |
| INTP3 | FCLA0CTL3_INTPL |
| INTP4 | FCLA0CTL4_INTPL |
| INTP5 | FCLA0CTL5_INTPL |
| INTP6 | FCLA0CTL6_INTPL |
| INTP7 | FCLA0CTL7_INTPL |
| INTP8 | FCLA0CTL0_INTPH |
| INTP9*1 | FCLA0CTL1_INTPH |
| INTP10 | FCLA0CTL2_INTPH |
| INTP11 | FCLA0CTL3_INTPH |
| INTP12 | FCLA0CTL4_INTPH |
| INTP13 | FCLA0CTL5_INTPH |
| INTP14*1 | FCLA0CTL6_INTPH |
| INTP15*1 | FCLA0CTL7_INTPH |

Note 1. For the supported products, see **Table 7.4, EI Level Maskable Interrupt Sources**.

See **Section 2, Pin Function** for details of these registers.

## 7.4 Interrupt Controller Control Registers

Writing to the ICxxx, IMRm (m = 0 to 11), FNC, and FIC registers is enabled only in supervisor mode (PSW.UM = 0).

### 7.4.1 List of Registers

Interrupt Controller Control Registers are listed in the following table.

**Table 7.12 List of Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| INTC1 | EI Level Interrupt Control Registers (Channel No.8 to 31) | ICxxx (see **Table 7.4**) | see **Table 7.4**. |
| INTC2 | EI Level Interrupt Control Registers (Channel No.32 to 357) | ICxxx (see **Table 7.4**) | see **Table 7.4**. |
| INTC1 | EI Level Interrupt Mask Registers (m = 0) | IMR0 | FFFE EAF0$_H$ |
| INTC2 | EI Level Interrupt Mask Registers (m = 1 to 11) | IMRm | FFFF B400$_H$ + (04$_H$ × m) |
| INTC1 | FE Level NMI Status Register | FNC | FFFE EA78$_H$ |
| | FE Level Maskable Interrupt Status Register | FIC | FFFE EA7A$_H$ |

## 7.4.2 ICxxx — EI Level Interrupt Control Registers

One of these registers is assigned to each EI level maskable interrupt (EIINT) channel and is used to set the conditions for controlling that channel. This register is initialized by any reset. For each source, see **Table 7.4, EI Level Maskable Interrupt Sources**.

**CAUTION**

**If 0 is written to the RFxxx bit immediately after a peripheral module generates the corresponding interrupt request in edge detection mode (before an interrupt is accepted by the CPU), the request may be lost.**

**Writing 1 to the RFxxx bit immediately after an interrupt is accepted by the CPU may lead to re-issuing of the request.**

**Writing method to these registers contains the use of bit manipulation instructions (set1, clr1, and not1).**

**For bit-manipulation instructions, see also Section 3.3.3, Access to Registers by Using Bit-Manipulation Instructions.**

**Executing a bit-manipulation instruction to the lower bytes including the MKxxx bit has no effect on the RFxxx bit.**

**Access:** ICxxx can be read or written in 16-bit units.
ICxxxH and ICxxxL can be read or written in 8- or 1-bit units.
Access to bits 14, 13, 11 to 8, 5, and 4 by using a SET1, CLR1, or NOT1 instruction is prohibited.

**Address:** See **Table 7.4, EI Level Maskable Interrupt Sources**.

**Value after reset:** $008F_H$ (edge detection), $808F_H$ (level detection)*1

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CTxxx | — | — | RFxxx | — | — | — | — | MKxxx | TBxxx | — | — | P3xxx | P2xxx | P1xxx | P0xxx |
| Value after reset | 0/1*1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R/W | R | R | R | R | R/W | R/W | R | R | R/W | R/W | R/W | R/W |

Note 1. The value after reset differs depending on the detection type of a given interrupt (edge detection: 0, level detection: 1). For details, see **Table 7.4, EI Level Maskable Interrupt Sources**.

**Table 7.13 ICxxx Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | CTxxx | This bit indicates the type of interrupt detection. This bit is read only.<br>0: Edge detection<br>1: Level detection<br>When writing in 8-bit or 16-bit units, write the value after reset. |
| 14, 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | RFxxx | This is an interrupt request flag.<br>The RFxxx bit can be written from a program. Setting the RFxxx bit to 1 generates an EI level maskable interrupt n (EIINTn), just as when an interrupt request is acknowledged.<br>0: No interrupt request is made.<br>1: Interrupt request is made.<br><br>**Input Interface** / **Operation**<br>Edge detection (CTxxx = 0) / This bit is automatically cleared when an interrupt request is acknowledged by the CPU core. It can be set and cleared by software.<br>Level detection (CTxxx = 1) / This bit cannot be set or cleared by software. It can only be read.<br>It is not cleared when an interrupt request is acknowledged by the CPU core. |
| 11 to 8 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

**Table 7.13 ICxxx Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | MKxxx | This is the interrupt request mask bit.<br>When the MKxxx bit is set, interrupt requests from the channel are masked and are not issued to the CPU core. The interrupt pending status is not reflected in the ICSR.PMEI bit for any channels that are masked.<br>When the interrupt request from the channel is masked with MKxxx = 1, the RFxxx still reflects the interrupt request for the channel and can be polled in software. When the MKxxx bit is cleared, interrupt requests from the channel are issued to the CPU core for subsequent processing. The state of the MKxxx bit is also reflected in the corresponding IMRm register.<br>0: Enables interrupt processing<br>1: Disables interrupt processing |
| 6 | TBxxx | This bit is used to select the way to determine the interrupt vector.<br>0: Direct jumping to an address determined from the level of priority<br>1: Table reference<br><br>For details on the way to determine the interrupt vector, see the *RH850G3KH User's Manual: Software*. |
| 5, 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 to 0 | P3xxx to P0xxx | These bits specify the interrupt priority as one of 16 levels, with 0 as the highest and 15 as the lowest.<br>When multiple EI level-interrupt requests are made simultaneously, the interrupt from the source with the highest priority setting in these bits is selected and conveyed to the CPU core for servicing first. When P3xxx to P0xxx bits specify the same priority level for simultaneously occurring interrupt requests, the source with the lower channel number takes priority. |

**CAUTION**

Do not access ICxxx registers of interrupt channels listed as "Reserved" in **Table 7.4, EI Level Maskable Interrupt Sources** and of the channels which are not incorporated in the product.

### 7.4.3 IMRm — EI Level Interrupt Mask Registers (m = 0 to 11)

These registers are a collection of the MKxxx bits of the ICxxx registers. Each bit of IMRm reflects the setting of the corresponding MKxxx bit. The setting for IMRm is also reflected in the corresponding MKxxx bit. This register is initialized by any reset.

**Access:** IMRm can be read or written in 32-bit units.
IMRmH and IMRmL can be read or written in 16-bit units.
IMRmHH, IMRmHL, IMRmLH, and IMRmLL can be read or written in 8- or 1-bit units.

**Address:** IMR0: FFFE EAF0$_H$
IMRm (m = 1 to 11): FFFF B400$_H$ + (04$_H$ × m)

**Value after reset:** FFFF FFFF$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IMRmEIMK(m × 32 + 31) | IMRmEIMK(m × 32 + 30) | IMRmEIMK(m × 32 + 29) | IMRmEIMK(m × 32 + 28) | IMRmEIMK(m × 32 + 27) | IMRmEIMK(m × 32 + 26) | IMRmEIMK(m × 32 + 25) | IMRmEIMK(m × 32 + 24) | IMRmEIMK(m × 32 + 23) | IMRmEIMK(m × 32 + 22) | IMRmEIMK(m × 32 + 21) | IMRmEIMK(m × 32 + 20) | IMRmEIMK(m × 32 + 19) | IMRmEIMK(m × 32 + 18) | IMRmEIMK(m × 32 + 17) | IMRmEIMK(m × 32 + 16) |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IMRmEIMK(m × 32 + 15) | IMRmEIMK(m × 32 + 14) | IMRmEIMK(m × 32 + 13) | IMRmEIMK(m × 32 + 12) | IMRmEIMK(m × 32 + 11) | IMRmEIMK(m × 32 + 10) | IMRmEIMK(m × 32 + 9) | IMRmEIMK(m × 32 + 8) | IMRmEIMK(m × 32 + 7) | IMRmEIMK(m × 32 + 6) | IMRmEIMK(m × 32 + 5) | IMRmEIMK(m × 32 + 4) | IMRmEIMK(m × 32 + 3) | IMRmEIMK(m × 32 + 2) | IMRmEIMK(m × 32 + 1) | IMRmEIMK(m × 32 + 0) |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.14 IMRm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | IMRmEIMK (m × 32 + 31) to IMRmEIMK (m × 32 + 0) | These are interrupt mask bits for EI level maskable interrupt (EIINT) channels 0 to 357.<br>0: Enables interrupt servicing<br>1: Disables interrupt servicing |

**CAUTION**

MKxxx bits which correspond to channels listed as “Reserved” in **Table 7.4, EI Level Maskable Interrupt Sources**, and to channels which are not incorporated in the product must be set to “1”.

### 7.4.4 FNC — FE Level NMI Status Register

This register indicates the status of an FE level non-maskable interrupt (FENMI).

**Access:** FNC is a read-only register that can be read in 16-bit units.
FNCH is a read-only register that can be read in 8- or 1-bit units.

**Address:** FNC: FFFE EA78$_H$
FNCH: FFFE EA79$_H$

**Value after reset:** 0000$_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | FNRF | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.15 FNC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 13 | Reserved | When read, the value after reset is returned. |
| 12 | FNRF | Interrupt request flag<br>0: No interrupt request<br>1: Interrupt request occurred<br><br>This bit is automatically cleared when an FE level NMI interrupt request is acknowledged by the CPU core. |
| 11 to 0 | Reserved | When read, the value after reset is returned. |

### 7.4.5 FIC — FE Level Maskable Interrupt Status Register

This register indicates the status of an FE level maskable interrupt (FEINT).

**Access:** FIC is a read-only register that can be read in 16-bit units.
FICH is a read-only register that can be read in 8- or 1-bit units.

**Address:** FIC: FFFE EA7A$_H$
FICH: FFFE EA7B$_H$

**Value after reset:** 8000$_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | FIRF | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.16 FIC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 13 | Reserved | When read, the value after reset is returned. |
| 12 | FIRF | Interrupt request flag<br>0: No interrupt request<br>1: Interrupt request occurred<br><br>This bit cannot be set or cleared by software. It can only be read. |
| 11 to 0 | Reserved | When read, the value after reset is returned. |

## 7.5 EI Level Maskable Interrupt Select Registers

The following registers are used to select an EI level maskable interrupt.

### 7.5.1 List of Registers

Interrupt select registers are listed in the following table.

**Table 7.17 List of Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| SL_INTC | INTC1 Interrupt Select Register | SELB_INTC1 | FFC0 $1000_H$ |
| | INTC2 Interrupt Select Register | SELB_INTC2 | FFC0 $1004_H$ |

## 7.5.2 SELB_INTC1 — INTC1 Interrupt Select Register

When two interrupt sources are assigned to one interrupt channel, this register selects which interrupt sources is enabled.

**NOTE**

The channel described in each bit setting indicates the channel of an interrupt and the priority. For details on channels, see **Table 7.4, EI Level Maskable Interrupt Sources**.

**Access:** This register can be read or written in 16-bit units.

**Address:** FFC0 $1000_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | SELB_INTC1_12 | SELB_INTC1_11 | SELB_INTC1_10 | SELB_INTC1_9 | SELB_INTC1_8 | SELB_INTC1_7 | SELB_INTC1_6 | SELB_INTC1_5 | SELB_INTC1_4 | SELB_INTC1_3 | SELB_INTC1_2 | SELB_INTC1_1 | SELB_INTC1_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.18 SELB_INTC1 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | SELB_INTC1_12 | Interrupt channel selection<br>0: INTADCA0I2 (Channel 20)<br>INTCSIH0IJC (Channel 32)<br>1: INTCSIH0IJC_1 (Channel 20)<br>INTADCA0I2_2 (Channel 32) |
| 11 | SELB_INTC1_11 | Interrupt channel selection<br>0: INTTAUD0I14 (Channel 15)<br>INTCSIH3IJC (Channel 161)<br>1: INTCSIH3IJC_1 (Channel 15)<br>INTTAUD0I14_2 (Channel 161) |
| 10 | SELB_INTC1_10 | Interrupt channel selection<br>0: INTTAUD0I12 (Channel 14)<br>INTCSIH3IRE (Channel 160)<br>1: INTCSIH3IRE_1 (Channel 14)<br>INTTAUD0I12_2 (Channel 160) |
| 9 | SELB_INTC1_9 | Interrupt channel selection<br>0: INTTAUD0I10 (Channel 13)<br>INTCSIH3IR (Channel 159)<br>1: INTCSIH3IR_1 (Channel 13)<br>INTTAUD0I10_2 (Channel 159) |
| 8 | SELB_INTC1_8 | Interrupt channel selection<br>0: INTTAUD0I2 (Channel 9)<br>INTCSIH3IC (Channel 158)<br>1: INTCSIH3IC_1 (Channel 9)<br>INTTAUD0I2_2 (Channel 158) |
| 7 | SELB_INTC1_7 | Interrupt channel selection<br>0: INTTAUD0I8 (Channel 12)<br>INTCSIH2IJC (Channel 135)<br>1: INTCSIH2IJC_1 (Channel 12)<br>INTTAUD0I8_2 (Channel 135) |
| 6 | SELB_INTC1_6 | Interrupt channel selection<br>0: INTTAUD0I6 (Channel 11)<br>INTCSIH2IRE (Channel 134)<br>1: INTCSIH2IRE_1 (Channel 11)<br>INTTAUD0I6_2 (Channel 134) |

Table 7.18 SELB_INTC1 Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 5 | SELB_INTC1_5 | Interrupt channel selection<br>0: INTTAUD0I4 (Channel 10)<br>INTCSIH2IR (Channel 133)<br>1: INTCSIH2IR_1 (Channel 10)<br>INTTAUD0I4_2 (Channel 133) |
| 4 | SELB_INTC1_4 | Interrupt channel selection<br>0: INTTAUD0I0 (Channel 8)<br>INTCSIH2IC (Channel 132)<br>1: INTCSIH2IC_1 (Channel 8)<br>INTTAUD0I0_2 (Channel 132) |
| 3 | SELB_INTC1_3 | Interrupt channel selection<br>0: INTCSIG0IR (Channel 28)<br>INTCSIH1IJC (Channel 119)<br>1: INTCSIH1IJC_1 (Channel 28)<br>INTCSIG0IR_2 (Channel 119) |
| 2 | SELB_INTC1_2 | Interrupt channel selection<br>0: INTCSIG0IC (Channel 27)<br>INTCSIH1IRE (Channel 118)<br>1: INTCSIH1IRE_1 (Channel 27)<br>INTCSIG0IC_2 (Channel 118) |
| 1 | SELB_INTC1_1 | Interrupt channel selection<br>0: INTTAPA0IVLY0 (Channel 17)<br>INTCSIH1IR (Channel 117)<br>1: INTCSIH1IR_1 (Channel 17)<br>INTTAPA0IVLY0_2 (Channel 117) |
| 0 | SELB_INTC1_0 | Interrupt channel selection<br>0: INTTAPA0IPEK0 (Channel 16)<br>INTCSIH1IC (Channel 116)<br>1: INTCSIH1IC_1 (Channel 16)<br>INTTAPA0IPEK0_2 (Channel 116) |

**CAUTION**

The operation of peripheral functions should be enabled after setting the corresponding interrupt source by SELB_INTC1.

### 7.5.3 SELB_INTC2 — INTC2 Interrupt Select Register

When two interrupt sources are assigned to one interrupt channel, this register selects which interrupt sources is enabled.

**Access:** This register can be read or written in 16-bit units.

**Address:** FFC0 $1004_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SELB_INTC2_10 | SELB_INTC2_9 | SELB_INTC2_8 | SELB_INTC2_7 | SELB_INTC2_6 | SELB_INTC2_5 | SELB_INTC2_4 | SELB_INTC2_3 | SELB_INTC2_2 | SELB_INTC2_1 | SELB_INTC2_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.19 SELB_INTC2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 11 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 10 | SELB_INTC2_10 | Interrupt channel 157 selection<br>0: INTTAUB0I15<br>1: INTPWGA31 |
| 9 | SELB_INTC2_9 | Interrupt channel 155 selection<br>0: INTTAUB0I13<br>1: INTPWGA30 |
| 8 | SELB_INTC2_8 | Interrupt channel 153 selection<br>0: INTTAUB0I11<br>1: INTPWGA26 |
| 7 | SELB_INTC2_7 | Interrupt channel 151 selection<br>0: INTTAUB0I9<br>1: INTPWGA19 |
| 6 | SELB_INTC2_6 | Interrupt channel 149 selection<br>0: INTTAUB0I7<br>1: INTPWGA18 |
| 5 | SELB_INTC2_5 | Interrupt channel 147 selection<br>0: INTTAUB0I5<br>1: INTPWGA17 |
| 4 | SELB_INTC2_4 | Interrupt channel 145 selection<br>0: INTTAUB0I3<br>1: INTPWGA16 |
| 3 | SELB_INTC2_3 | Interrupt channel 88 selection<br>0: INTENCA0I1<br>1: INTPWGA7 |
| 2 | SELB_INTC2_2 | Interrupt channel 87 selection<br>0: INTENCA0I0<br>1: INTPWGA6 |
| 1 | SELB_INTC2_1 | Interrupt channel 86 selection<br>0: INTENCA0IUD<br>1: INTPWGA5 |
| 0 | SELB_INTC2_0 | Interrupt channel 85 selection<br>0: INTENCA0IOV<br>1: INTPWGA4 |

**CAUTION**

The operation of peripheral functions should be enabled after setting the corresponding interrupt source by SELB_INTC2.

## 7.6 Interrupt Function System Registers

See **Table 3.30, Interrupt Function System Registers**.

### 7.6.1 FPIPR — FPI Exception Interrupt Priority

See **Table 3.31, FPIPR Register Contents**.

### 7.6.2 ISPR — Priority of Interrupt being Serviced

See **Table 3.32, ISPR Register Contents**.

### 7.6.3 PMR — Interrupt Priority Masking

See **Table 3.33, PMR Register Contents**.

### 7.6.4 ICSR — Interrupt Control Status

See **Table 3.34, ICSR Register Contents**.

### 7.6.5 INTCFG — Interrupt Function Setting

See **Table 3.35, INTCFG Register Contents**.

## 7.7 Operation when Acknowledging an Interrupt

Check whether each interrupt that is reported during instruction execution is acknowledged according to the priority. The procedure for acknowledging each interrupt is shown below.

(1) Check whether the acknowledgment conditions are satisfied and whether interrupts are acknowledged according to their priority.

(2) Calculate the exception handler address according to the current PSW value.*1

(3) For FE-level non-maskable/maskable interrupts, the following processing is performed:

- Save the PC to the FEPC.
- Save the PSW to the FEPSW.
- Store the exception source code in the FEIC.
- Update the PSW and MCTL.*2
- Store the exception handler address calculated in (2) in the PC, and then pass its control to the exception handler.

(4) For EI level exceptions, the following processing is performed:

- Save the PC to the EIPC.
- Save the PSW to the EIPSW.
- Store the exception source code in the EIIC.
- Update the PSW and MCTL.*2
- Store the exception handler address calculated in (2) in the PC, and then pass its control to the exception handler.

**Note 1.** For details, see **Section 7.10, Exception Handler Address**.

**Note 2.** For the values to be updated, see *Table 4.1 Exception Cause List* in the *RH850G3KH User's Manual: Software*.

The following figure shows steps (1) to (4).

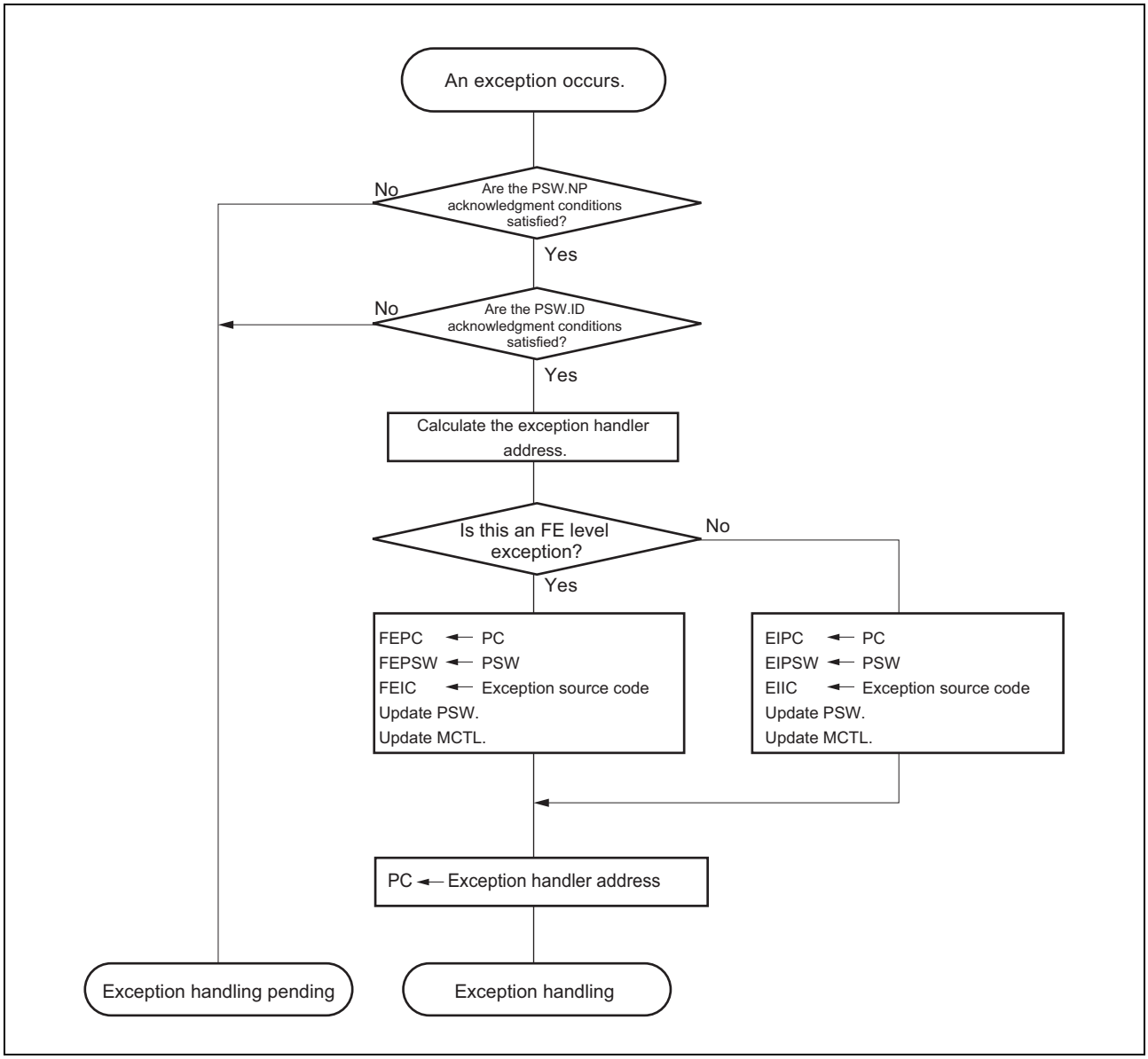



**Figure 7.2 Operation when Acknowledging an Interrupt**

### 7.7.1 Exception Source Codes for Different Types of SYSERR Exceptions

The following table lists exception source codes for the different types of SYSERR exceptions.

**Table 7.20 Exception Source Codes for Different Types of SYSERR Exceptions**

| Exception Source Code | Source of SYSERR Generation |
|---|---|
| $11_H$ | • Detection of an error during the fetching of an instruction from the code flash memory area |
| $13_H$ | • Detection of an error during the fetching of an instruction from the local or retention RAM areas |
| $14_H$ | • Detection of an error during access to data in the code flash area<br>• Detection of an error during read access to a module via the system interconnect or PBUS<br>The exception source code reports an occurrence of a system error factor corresponding to VCIF bit of SEGFLAG register*1 |
| $16_H$ | • Detection of an error during access to data in the local or retention RAM areas<br>The exception source code reports an occurrence of a system error factor corresponding to TCMF bit of SEGFLAG register*1 |
| $19_H$ | • Detection of an error during write access to a module via the PBUS<br>The exception source code reports an occurrence of a system error factor corresponding to APIF bit of SEGFLAG register*1 |
| $1A_H$ | • Detection of an error inside system interconnect<br>The exception source code reports an occurrence of a system error factor corresponding to VCSF bit of SEGFLAG register*1 |

Note 1. See **Section 3.2.3.2, System Error Generator Function (SEG)** for details.

## 7.8 Return from Interrupts

To return from interrupt handling, execute the return instruction (EIRET or FERET) corresponding to each relevant interrupt level.

When a context has been saved in a stack and the like, the context must be restored before executing the return instruction.

The EIRET instruction is used to return from the EI level maskable interrupt handling and the FERET instruction is used to return from FE-level maskable interrupt handling.

When the EIRET or FERET instruction is executed, the CPU performs the following processing and then passes its control to the return PC address:

(1) When returning from the service routine for an EI-level exception, the PC and PSW values on return are loaded from the EIPC and EIPSW registers.
When returning from the service routine for an FE-level exception, the PC and PSW values on return are loaded from the FEPC and FEPSW registers.

(2) Control is passed to the addresses indicated by the return PC and PSW that were loaded.

(3) When EP = 0 and INTCFG.ISPC = 0, the CPU updates the ISPR register.

The flows for returning from exception handling using the EIRET and FERET instructions are shown below.

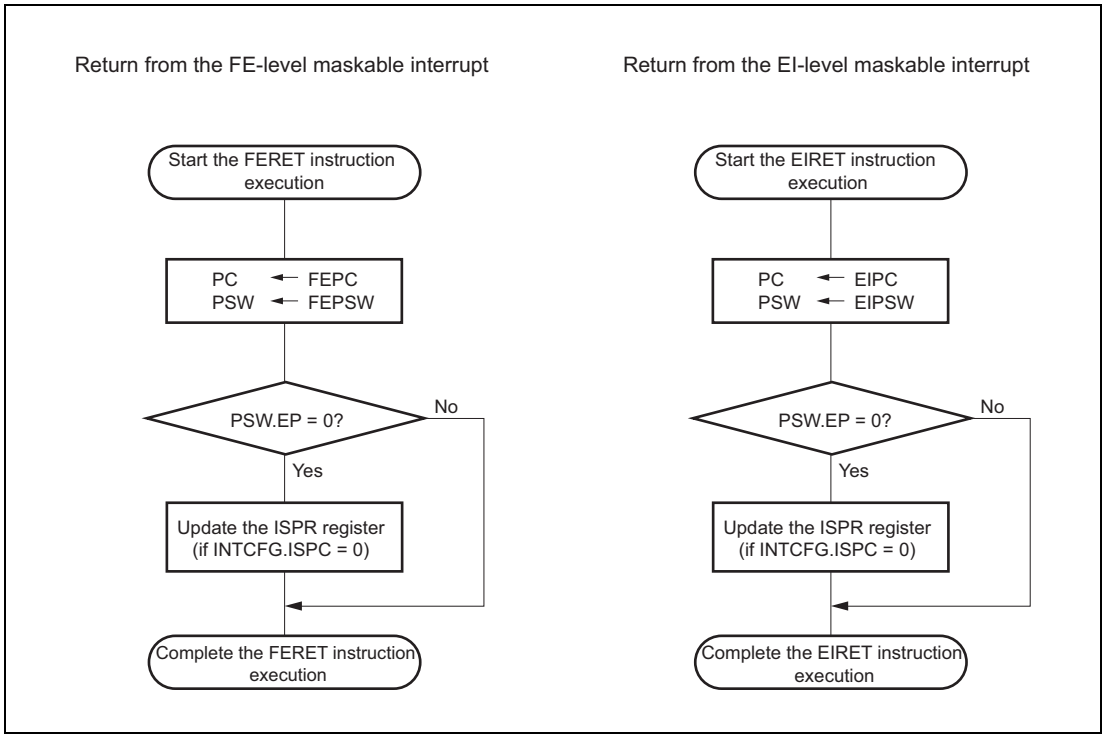



**Figure 7.3 Flow of Return from Interrupts**

## 7.9 Interrupt Operation

### 7.9.1 Interrupt Mask Function of EI Level Maskable Interrupt (EIINT)

Interrupt masking can be specified for each respective interrupt channel of EIINT. Interrupt masking is performed by the following register settings.

**Table 7.21 Operation of the MKxxx Bit**

| ICxxx.MKxxx | Operation |
|---|---|
| 1 | Masks interrupt |
| 0 | Enables interrupt |

The ICxxx.MKxxx bits can also be read and written via the corresponding IMRmEIMKn bits of the IMRm registers. The interrupt mask state is reflected in both the ICxxx registers and the IMRm registers.

**[ Operation example ]**

(1) When a 1 is written to an IMRm.IMRmEIMKn bit, interrupts are prohibited for the corresponding channel.

(2) When the corresponding ICxxx.MKxxx bit is read, 1 is returned.

**CAUTION**

**If the MKxxx bit is set to 0 while an interrupt request is pending (RFxxx = 1), the interrupt service routine will be executed at that time (subject to the rules of interrupt prioritization). Even if an interrupt request is issued in software by setting the RFxxx bit to 1, the interrupt will not occur as long as the interrupt is masked with MKxxx = 1.**

**To cancel an interrupt request that is pending, clear the corresponding RFxxx bit in software.**

### 7.9.2 Interrupt Priority Level Judgment

When FE level non-maskable interrupts (FENMI), FE level maskable interrupts (FEINT), and EI level maskable interrupts (EIINT) are input, priorities including other exceptions are determined, and the exception with the highest priority (including interrupts) is processed. Exceptions occurred at the same time (including interrupts) are processed in a pre-allocated priority order (the default priority order). The priority orders of FENMI, FEINT, and EIINT interrupts are as follows.

FENMI > FEINT > EIINT

See the *RH850G3KH User's Manual: Software* for other exceptions.

For EIINT(INTn) interrupts, the interrupt priority can be set independently for each interrupt source. Specify the interrupt priority with the bits P3xxx to P0xxx. The interrupt priority levels can be set from 0 to 15: 0 is the highest and 15 is the lowest. Among multiple EIINT(INTn) interrupts with the same priority level, the interrupt with the lowest interrupt channel number has priority.

Table 7.22 Example of EIINT (INTn) Interrupt Priority Level Settings and Priority Levels During Operation

| EIINT(INTn) | ICxxx.P[3:0]xxx Setting | Priority Level During Operation |
|---|---|---|
| INT0 | 3 | 10 |
| INT1 | 4 | 11 |
| INT2 | 0 | 1 |
| INT3 | 0 | 2 |
| INT4 | 1 | 3 |
| INT5 | 2 | 6 |
| INT6 | 2 | 7 |
| INT7 | 1 | 4 |
| INT8 | 1 | 5 |
| INT9 | 2 | 8 |
| INT10 | 2 | 9 |

The interrupt controller executes multiple interrupt handling when another interrupt request is acknowledged while an interrupt processing is being executed. When multiple EIINT(INTn) interrupts occur at the same time, the interrupt to be acknowledged is determined by the following procedure.

#### 7.9.2.1 Comparison with the Priority Level of the Interrupt Currently being Handled

Interrupts with the same or lower priority level as the interrupt currently being handled are held pending.

The priority level of the interrupt currently being handled is stored in the ISPR register.

Interrupts with a higher priority level than the interrupt currently being handled proceed to the next priority judgment stage.

#### 7.9.2.2 Masking through Priority Mask Register (PMR)

Only interrupts enabled by the PMR register proceed to the next priority judgment stage.

For the PMR register, see **Table 3.33, PMR Register Contents**, or the *RH850G3KH User's Manual: Software*.

#### 7.9.2.3 The Requested Interrupt Source with the Highest Priority Level is Selected

When interrupts are requested simultaneously from multiple sources, the interrupt set the highest priority by ICxxx.P[3:0]xxx bits takes priority.

When there are multiple highest priority interrupts, the lowest interrupt channel number is selected.

#### 7.9.2.4 Interrupt Suspended by CPU

Interrupt acknowledgment is held according to the state of the NP and ID bits of the PSW register. At this time, priority judgment among EIINT interrupts, and priority judgment among EIINT, FEINT and FENMI interrupts are performed even while interrupt acknowledgment is pending, and the interrupt with the highest priority is selected when the acknowledgment condition is satisfied.

**Example**

An EIINT interrupt with the priority level 5 has already been requested and interrupt generation is pending because the value of the PSW.ID bit is 1. If a subsequent EIINT interrupt with the priority

level 3 is requested and the PSW.ID bit is cleared to 0, the latter EIINT interrupt (with the priority level 3) will be generated.

**Figure 7.4** shows an example of multiple interrupt handling when another interrupt request is acknowledged while interrupt processing is being executed.

When an interrupt request signal is acknowledged, the PSW.ID flag is automatically set to 1. Therefore, the ID flag should be cleared to 0 to execute multiple interrupt handling. Specifically, execute the EI instruction and the like in an interrupt handling program to enable the interrupt.

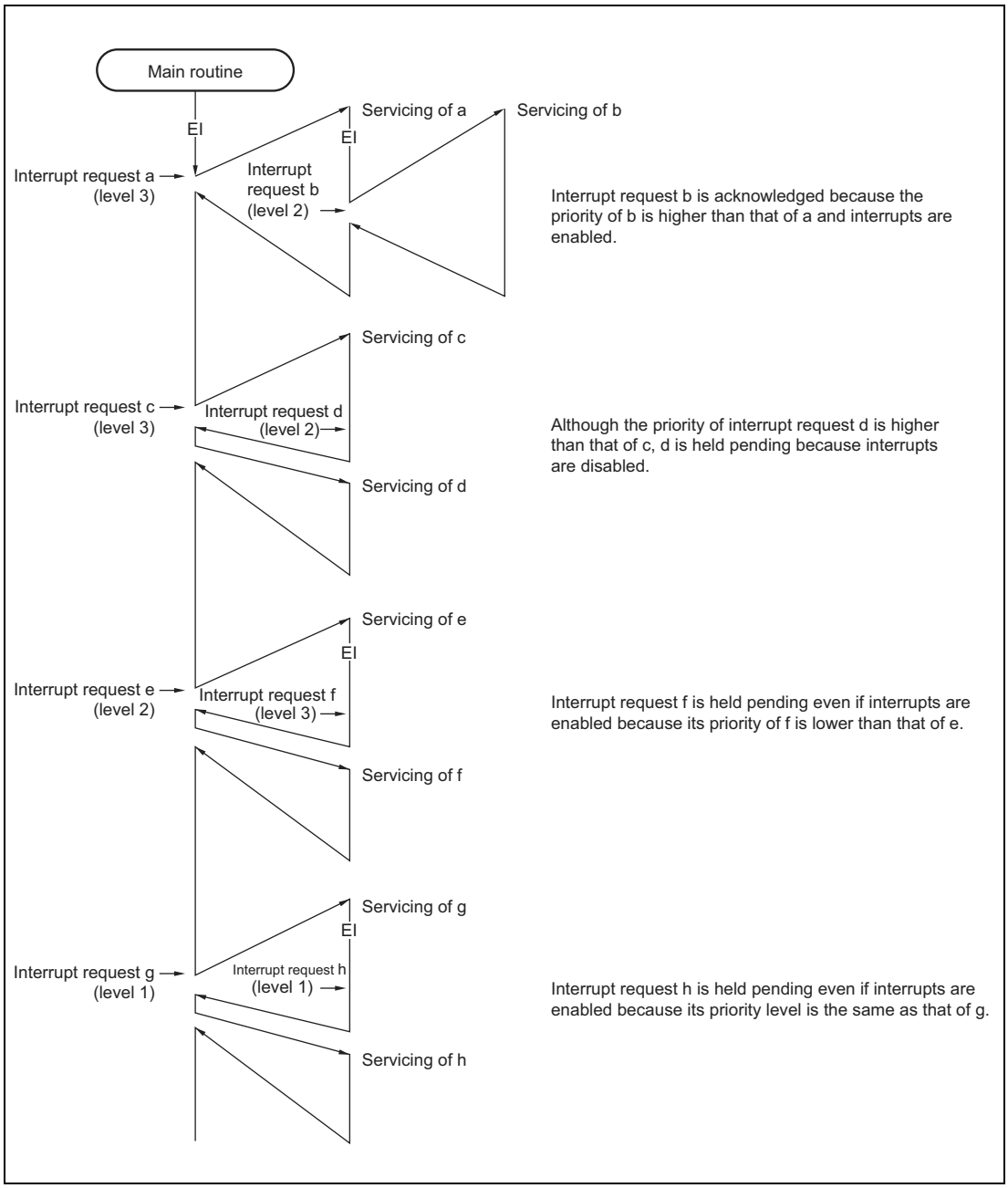
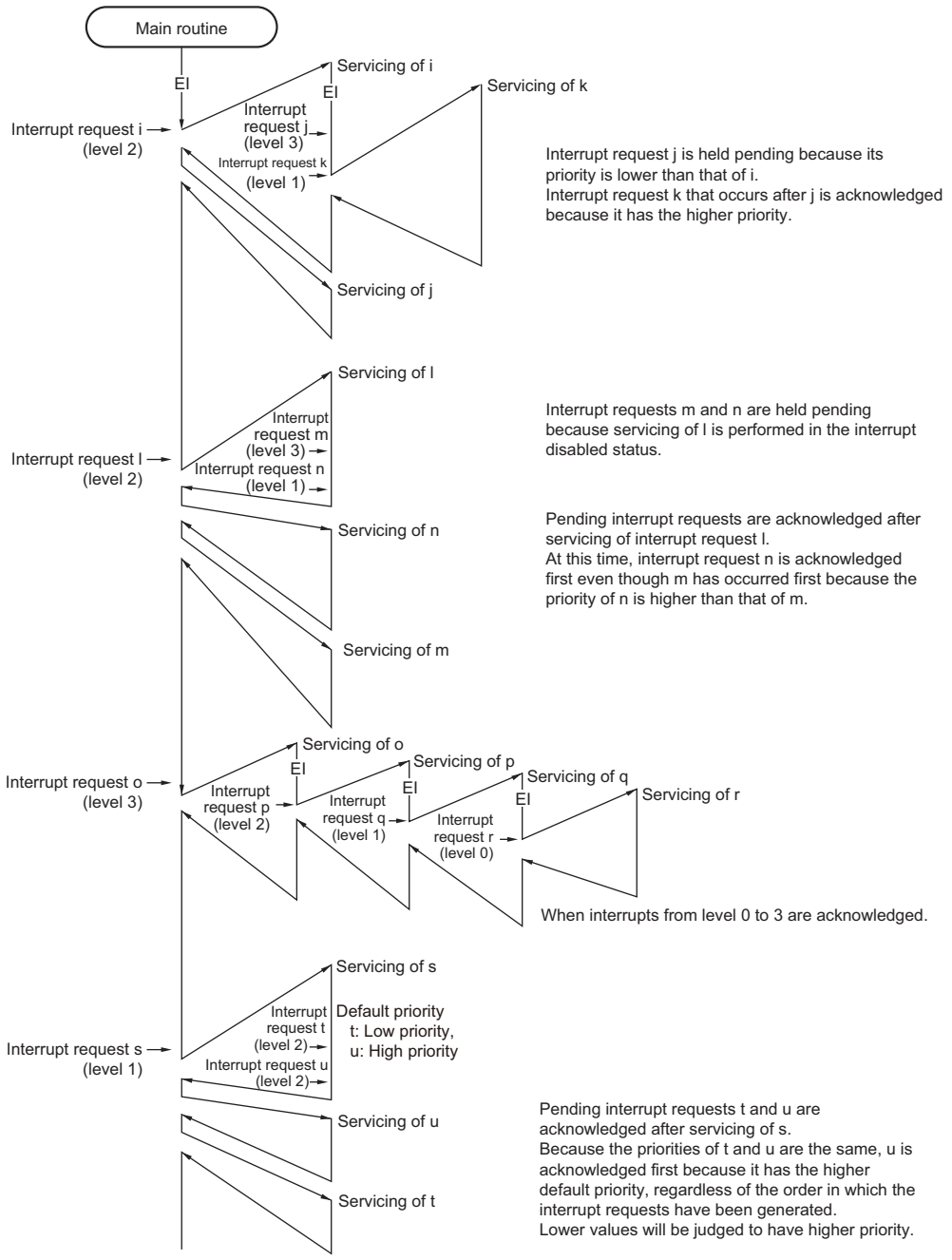



**Figure 7.4 Example of Processing in which an Interrupt Request Signal is Issued while Another Interrupt is being Handled (1)**



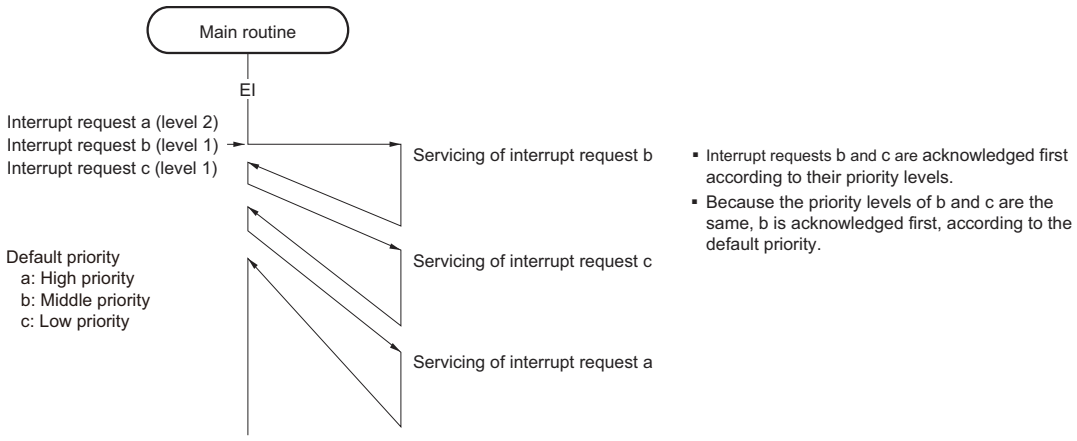

**NOTES**

1. “a” to “u” in the figure are the temporary names of interrupt request signals shown for the sake of explanation.
2. The default priority (high/low) in the figure indicates the relative priority between two interrupt request signals.

**Figure 7.5 Example of Processing in which an Interrupt Request Signal is Issued while Another Interrupt is being Handled (2)**

**CAUTION**

**To perform multiple interrupt servicing, the values of the EIPC and EIPSW registers must be saved before executing the EI instruction. When returning from multiple interrupt servicing, restore the values of EIPC and EIPSW after executing the DI instruction.**




**NOTES**

1. “a” to “u” in the figure are the temporary names of interrupt request signals shown for the sake of explanation.
2. The default priority in the figure indicates the relative priority between two interrupt request signals.

**Figure 7.6 Example of Servicing Simultaneously Generated Interrupt Requests**

### 7.9.3 Interrupt Request Acknowledgement Conditions and the Priority

See the *RH850G3KH User's Manual: Software*.

### 7.9.4 Exception Priority of Interrupts and the Priority Mask

See the *RH850G3KH User's Manual: Software*.

### 7.9.5 Interrupt Priority Mask

See the *RH850G3KH User's Manual: Software*.

### 7.9.6 Priority Mask Function

The priority mask function prohibits all EIINT interrupts of the specified interrupt priority level.

The interrupt priority levels to be masked are specified in the PMR register. Masking and acknowledgment can be set for each priority level.

The following operations are possible using this function:

- Temporary prohibition of interrupts that have a priority level that is lower than a given priority level
- Temporary prohibition of interrupts that have a given priority level

**Table 7.23 Operation of the PMR.PMm Bit**

| PMR.PMm | Operation |
|---|---|
| 0 | Acknowledges requests from priority level m interrupt source. |
| 1 | Masks requests from priority level m interrupt source. |

**Note:** m = 0 to 15

The presence of EIINT interrupts held pending with this function can be checked with **Section 7.9.7, Exception Management**.

For details on the PMR register, see **Table 3.33, PMR Register Contents**, or the *RH850G3KH User's Manual: Software*.

### 7.9.7 Exception Management

Pending interrupts can be checked in the RH850/F1K. For details, see the *RH850G3KH User's Manual: Software*.

## 7.10 Exception Handler Address

In the RH850/F1K, the exception handler address from which the handler is executed after a reset is input or when an exception or interrupt is acknowledged can be changed according to a setting.

The exception handler address for resets and exceptions (including interrupts) is determined with the direct vector method, in which the reference point of the exception handler address can be changed by using the PSW.EBV bit, the RBASE register, and the EBASE register. For interrupts, the direct vector method and table reference method can be selected for each channel. If the table reference method is selected, execution can branch to the address indicated by the exception handler table allocated in the memory.

**CAUTION**

**The exception handler address of EIINT(INTn) selected using the direct vector method differs from that of the V850E2 core products. In the V850E2 core products, a different exception handler address is individually assigned to each interrupt channel (EIINT(INTn)). In the RH850/F1K, one exception handler address is assigned to each interrupt priority. Consequently, interrupts that have the same priority level branch to the same exception handler.**

### 7.10.1 Direct Vector Method

The CPU uses the result of adding the offset shown in **Table 7.24, Selection of Base Register/ Offset Address** to the base address indicated by the RBASE or EBASE register as the exception handler address.

Select whether the RBASE or EBASE register is used as the base address by using the PSW.EBV bit*1. When the PSW.EBV bit is set to 1, the value of the EBASE register is used as the base address. When the PSW.EBV bit is cleared to 0, the value of the RBASE register is used as the base address.

For reset input and some exceptions*2, however, the RBASE register is always used for reference.

In addition, user interrupts see the RINT bit of the corresponding base register, and reduce the offset address according to the bit status. If the RBASE.RINT bit or EBASE.RINT bit is set to 1, all user interrupts are handled using an offset of $100_H$. If the bit is cleared to 0, the offset address is determined according to **Table 7.24, Selection of Base Register/Offset Address**.

**Note 1.** Exception acknowledgment itself may sometimes update the status of the PSW.EBV bit. In this case, the base register is selected based on the updated value.

**Note 2.** The exceptions that always see the RBASE register are determined according to the hardware specifications.

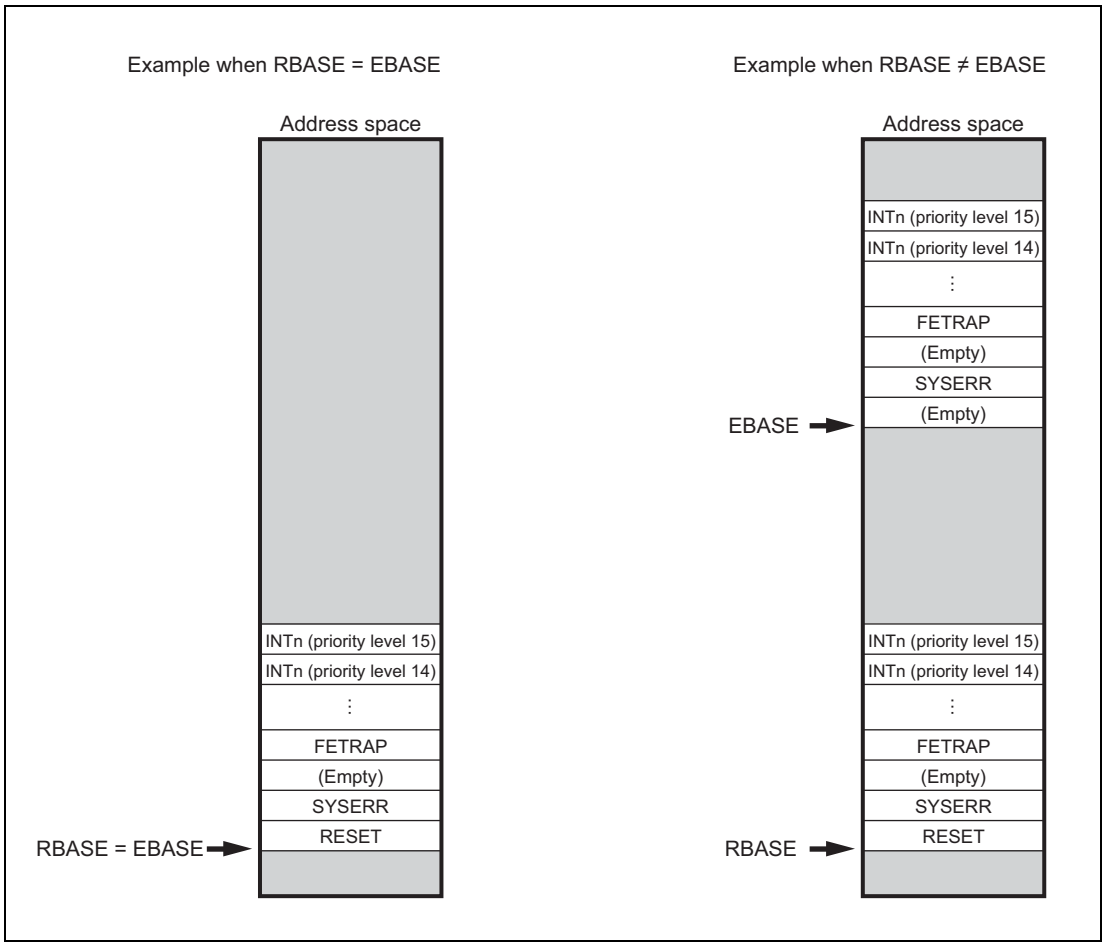



**Figure 7.7 Direct Vector Method**

The table below shows how base register selection and offset address reduction function for each exception to determine the exception handler address. The value of the PSW bit determines the exception handler address on the basis of the value updated by the acknowledgment of an exception.

**Table 7.24 Selection of Base Register/Offset Address**

| Exception/Interrupt | PSW.EBV = 0 | PSW.EBV = 1 | RINT = 0 | RINT = 1 |
|---|---|---|---|---|
| | Base Register | | Offset Address | |
| RESET | RBASE | N.A. | $000_H$ | $000_H$ |
| SYSERR | | EBASE | $010_H$ | $010_H$ |
| Reserved | | | $020_H$ | $020_H$ |
| FETRAP | | | $030_H$ | $030_H$ |
| TRAP0 | | | $040_H$ | $040_H$ |
| TRAP1 | | | $050_H$ | $050_H$ |
| RIE | | | $060_H$ | $060_H$ |
| FPP/FPI | | | $070_H$ | $070_H$ |
| UCPOP | | | $080_H$ | $080_H$ |
| MIP/MDP | | | $090_H$ | $090_H$ |
| PIE | | | $0A0_H$ | $0A0_H$ |
| Debug | | | $0B0_H$ | $0B0_H$ |
| MAE | | | $0C0_H$ | $0C0_H$ |
| Reserved | | | $0D0_H$ | $0D0_H$ |
| FENMI | | | $0E0_H$ | $0E0_H$ |
| FEINT | | | $0F0_H$ | $0F0_H$ |
| INTn (Priority level 0) | | | $100_H$ | $100_H$ |
| INTn (Priority level 1) | | | $110_H$ | |
| INTn (Priority level 2) | | | $120_H$ | |
| INTn (Priority level 3) | | | $130_H$ | |
| INTn (Priority level 4) | | | $140_H$ | |
| INTn (Priority level 5) | | | $150_H$ | |
| INTn (Priority level 6) | | | $160_H$ | |
| INTn (Priority level 7) | | | $170_H$ | |
| INTn (Priority level 8) | | | $180_H$ | |
| INTn (Priority level 9) | | | $190_H$ | |
| INTn (Priority level 10) | | | $1A0_H$ | |
| INTn (Priority level 11) | | | $1B0_H$ | |
| INTn (Priority level 12) | | | $1C0_H$ | |
| INTn (Priority level 13) | | | $1D0_H$ | |
| INTn (Priority level 14) | | | $1E0_H$ | |
| INTn (Priority level 15) | | | $1F0_H$ | |

Base register selection is used to execute exception handling for resets and some hardware errors by using the programs in a relatively reliable area such as ROM instead of the areas that are easily affected by software errors such as RAM and cache area. The user interrupt offset address reduction function is used to reduce the memory occupation size required by the exception handler for specific system-internal operating modes. The main purpose of this is to minimize the amount of memory consumed in operating modes that use only the minimum functionality, for example, during system maintenance and diagnosis.

### 7.10.2 Table Reference Method

With the direct vector method, there is one user-interrupt exception handler for each interrupt priority level, and interrupt channels that indicate multiple interrupts with the same priority branch to the same interrupt handler, but some users might want to use different code areas for each interrupt handler from the beginning.

The RH850/F1K uses the table reference method for interrupts that assume the above usage.

If the table reference method is specified as the interrupt channel vector selection method in the interrupt controller and the like, the method for determining the exception handler address when an interrupt request corresponding to that interrupt channel is acknowledged differs as follows:

<1> In any of the following cases, the exception handler address is determined by using the direct vector method:

- When PSW.EBV = 0 and RBASE.RINT = 1
- When PSW.EBV = 1 and EBASE.RINT = 1
- When the interrupt channel setting is not the table reference method

<2> In cases other than <1>, calculate the table reference position.
Exception handler address read position = INTBP register + channel number × 4 bytes

<3> Read word data starting at the interrupt handler address read position calculated in <2>.

<4> Use the word data read in <3> as the exception handler address.

**Table 7.25** shows the exception handler address read positions corresponding to each interrupt channel and **Figure 7.8** shows an overview of the allocation in memory.

**Table 7.25 Exception Handler Address Expansion**

| Type of Interrupt | Exception Handler Address Read Position |
|---|---|
| EI level maskable interrupt channel 0 | INTBP register value + 0 × 4 |
| EI level maskable interrupt channel 1 | INTBP register value + 1 × 4 |
| EI level maskable interrupt channel 2 | INTBP register value + 2 × 4 |
| : | : |
| EI level maskable interrupt channel 356 | INTBP register value + 356 × 4 |
| EI level maskable interrupt channel 357 | INTBP register value + 357 × 4 |

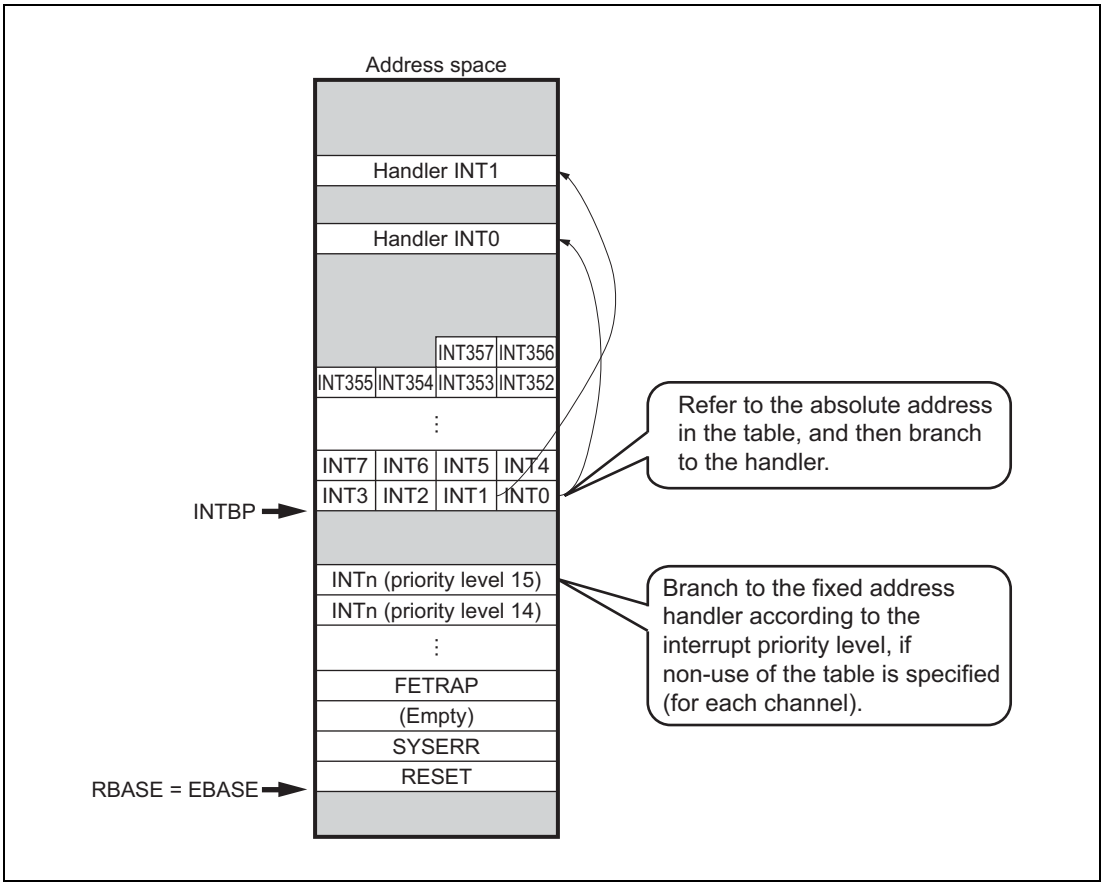



**Figure 7.8 Table Reference Method**

# Section 8 DMA Controller

This section explains about the DMA controller (DMA) in general.

The first part in this section describes the features specific to the RH850/F1K, including the number of channels and register base addresses. The ensuing sections describe the DMA functions and registers.

## 8.1 Features of RH850/F1K DMA Controller

### 8.1.1 Number of Channels

The products of the RH850/F1K series incorporate a DMA with the number of channels indicated below.

**Table 8.1 Number of Channels**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of channels | 16 ch (8ch × 2) | | |

**Table 8.2 Indices**

| Index | Description |
|---|---|
| n | In this section, the DMA units are identified with an index “n” (n = 0, 1) for example, as DMACn. |
| m | In this section, the DMA channels are identified with an index “m” (m = 0 to 15). For example, DSAm is the DMA source address register. |
| i | In this section, the DMAC channels are identified with an index “i” (i = 0 to 7). For example, DMniCM is the DMAC channel master. |

### 8.1.2 Register Base Address

The DMA base address is indicated in the following table.

The DMA register address is expressed as an offset of the base address.

**Table 8.3 Register Base Address**

| Base Address Name | Base Address |
|---|---|
| <DMA_base> | FFFF $8000_H$ |

### 8.1.3 Interrupt Requests

The DMA interrupt requests are listed in the following table.

**Table 8.4 Interrupt Requests**

| Unit Interrupt Signal | Description | Interrupt Number |
|---|---|---|
| INTDMA0 | DMA00 transfer completion | 60 |
| INTDMA1 | DMA01 transfer completion | 61 |
| INTDMA2 | DMA02 transfer completion | 62 |
| INTDMA3 | DMA03 transfer completion | 63 |
| INTDMA4 | DMA04 transfer completion | 64 |
| INTDMA5 | DMA05 transfer completion | 65 |
| INTDMA6 | DMA06 transfer completion | 66 |
| INTDMA7 | DMA07 transfer completion | 67 |
| INTDMA8 | DMA10 transfer completion | 68 |
| INTDMA9 | DMA11 transfer completion | 69 |
| INTDMA10 | DMA12 transfer completion | 70 |
| INTDMA11 | DMA13 transfer completion | 71 |
| INTDMA12 | DMA14 transfer completion | 72 |
| INTDMA13 | DMA15 transfer completion | 73 |
| INTDMA14 | DMA16 transfer completion | 74 |
| INTDMA15 | DMA17 transfer completion | 75 |

**Table 8.5 Interrupt Request (FE Level Non-Maskable Interrupts)**

| Unit Interrupt Signal | Description | Interrupt Name |
|---|---|---|
| INTDMAERR | DMA transfer error interrupt | INTDMAERR |

### 8.1.4 DMA Trigger Factors

DMA trigger factors can be selected by setting the DTFRm.REQSEL[6:0] bits.

The following table lists all DMA trigger factors which can be selected by the DTFRm register.

**Table 8.6 DMA Trigger Factors (1/4)**

| DMA Trigger Number DTFRm.REQSEL[6:0] | DMA Trigger Factor | F1K 100 pins | F1K 144 pins | F1K 176 pins |
|---|---|---|---|---|
| DMACTRG[0] | INTTAUD0I0 | √ | √ | √ |
| DMACTRG[1] | INTTAUD0I4 | √ | √ | √ |
| DMACTRG[2] | INTTAUD0I8 | √ | √ | √ |
| DMACTRG[3] | INTTAUD0I12 | √ | √ | √ |
| DMACTRG[4] | INTADCA0I0 | √ | √ | √ |
| DMACTRG[5] | INTADCA0I1 | √ | √ | √ |
| DMACTRG[6] | INTADCA0I2 | √ | √ | √ |
| DMACTRG[7] | ADC_CONV_END0 | √ | √ | √ |
| DMACTRG[8] | INTCSIG0IC | √ | √ | √ |
| DMACTRG[9] | INTCSIG0IR | √ | √ | √ |
| DMACTRG[10] | INTRLIN30UR0 | √ | √ | √ |
| DMACTRG[11] | INTRLIN30UR1 | √ | √ | √ |
| DMACTRG[12] | INTP0 | √ | √ | √ |
| DMACTRG[13] | INTP2 | √ | √ | √ |
| DMACTRG[14] | INTP4 | √ | √ | √ |
| DMACTRG[15] | INTTAUD0I1 | √ | √ | √ |
| DMACTRG[16] | INTTAUD0I5 | √ | √ | √ |
| DMACTRG[17] | INTTAUD0I9 | √ | √ | √ |
| DMACTRG[18] | INTTAUD0I13 | √ | √ | √ |
| DMACTRG[19] | INTRIIC0TI | √ | √ | √ |
| DMACTRG[20] | INTRIIC0RI | √ | √ | √ |
| DMACTRG[21] | INTTAUJ0I0 | √ | √ | √ |
| DMACTRG[22] | INTTAUJ0I3 | √ | √ | √ |
| DMACTRG[23] | RSCANFDCF0*[1] | √ | √ | √ |
| DMACTRG[24] | RSCANFDCF1*[1] | √ | √ | √ |
| DMACTRG[25] | Setting prohibited | — | — | — |
| DMACTRG[26] | RSCANFDCF2*[1] | √ | √ | √ |
| DMACTRG[27] | RSCANFDCF3*[1] | √ | √ | √ |
| DMACTRG[28] | INTCSIH1IC | √ | √ | √ |
| DMACTRG[29] | INTCSIH1IR | √ | √ | √ |
| DMACTRG[30] | INTCSIH1IJC | √ | √ | √ |
| DMACTRG[31] | INTP6 | √ | √ | √ |
| DMACTRG[32] | INTP8 | √ | √ | √ |
| DMACTRG[33] | INTTAUB0I0 | √ | √ | √ |
| DMACTRG[34] | INTTAUB0I2 | √ | √ | √ |
| DMACTRG[35] | INTTAUB0I4 | √ | √ | √ |
| DMACTRG[36] | INTTAUB0I6 | √ | √ | √ |
| DMACTRG[37] | INTTAUB0I9 | √ | √ | √ |
| DMACTRG[38] | INTTAUB0I11 | √ | √ | √ |

Table 8.6 DMA Trigger Factors (2/4)

| DMA Trigger Number DTFRm.REQSEL[6:0] | DMA Trigger Factor | F1K 100 pins | F1K 144 pins | F1K 176 pins |
|---|---|---|---|---|
| DMACTRG[39] | INTTAUB0I13 | √ | √ | √ |
| DMACTRG[40] | INTTAUB0I15 | √ | √ | √ |
| DMACTRG[41] | INTCSIH3IC | √ | √ | √ |
| DMACTRG[42] | INTCSIH3IR | √ | √ | √ |
| DMACTRG[43] | INTCSIH3IJC | √ | √ | √ |
| DMACTRG[44] | INTRLIN32UR0 | √ | √ | √ |
| DMACTRG[45] | INTRLIN32UR1 | √ | √ | √ |
| DMACTRG[46] | INTTAUJ1I0 | √ | √ | √ |
| DMACTRG[47] | INTTAUJ1I2 | √ | √ | √ |
| DMACTRG[48] | RSCANFDCF4*[1] | √ | √ | √ |
| DMACTRG[49] | RSCANFDCF5*[1] | √ | √ | √ |
| DMACTRG[50] | INTRLIN34UR0 | — | √ | √ |
| DMACTRG[51] | INTRLIN34UR1 | — | √ | √ |
| DMACTRG[52] | INTTAUB1I0 | — | — | √ |
| DMACTRG[53] | INTTAUB1I2 | — | — | √ |
| DMACTRG[54] | INTTAUB1I4 | — | — | √ |
| DMACTRG[55] | INTTAUB1I6 | — | — | √ |
| DMACTRG[56] | INTTAUB1I9 | — | — | √ |
| DMACTRG[57] | INTTAUB1I11 | — | — | √ |
| DMACTRG[58] | INTTAUB1I13 | — | — | √ |
| DMACTRG[59] | INTTAUB1I15 | — | — | √ |
| DMACTRG[60] | RSCANFDRF0*[1] | √ | √ | √ |
| DMACTRG[61] | RSCANFDRF1*[1] | √ | √ | √ |
| DMACTRG[62] | RSCANFDRF2*[1] | √ | √ | √ |
| DMACTRG[63] | RSCANFDRF3*[1] | √ | √ | √ |
| DMACTRG[64] | INTTAUD0I2 | √ | √ | √ |
| DMACTRG[65] | INTTAUD0I6 | √ | √ | √ |
| DMACTRG[66] | INTTAUD0I10 | √ | √ | √ |
| DMACTRG[67] | INTTAUD0I14 | √ | √ | √ |
| DMACTRG[68] | RSCANFDRF4*[1] | √ | √ | √ |
| DMACTRG[69] | RSCANFDRF5*[1] | √ | √ | √ |
| DMACTRG[70] | INTCSIH0IC | √ | √ | √ |
| DMACTRG[71] | INTCSIH0IR | √ | √ | √ |
| DMACTRG[72] | INTCSIH0IJC | √ | √ | √ |
| DMACTRG[73] | INTP1 | √ | √ | √ |
| DMACTRG[74] | INTP3 | √ | √ | √ |
| DMACTRG[75] | INTP5 | √ | √ | √ |
| DMACTRG[76] | INTTAUD0I3 | √ | √ | √ |
| DMACTRG[77] | INTTAUD0I7 | √ | √ | √ |
| DMACTRG[78] | INTTAUD0I11 | √ | √ | √ |
| DMACTRG[79] | INTTAUD0I15 | √ | √ | √ |
| DMACTRG[80] | INTTAUJ0I1 | √ | √ | √ |
| DMACTRG[81] | INTTAUJ0I2 | √ | √ | √ |

Table 8.6 DMA Trigger Factors (3/4)

| DMA Trigger Number DTFRm.REQSEL[6:0] | DMA Trigger Factor | F1K 100 pins | F1K 144 pins | F1K 176 pins |
|---|---|---|---|---|
| DMACTRG[82] | RSCANFDRF6*[1] | √ | √ | √ |
| DMACTRG[83] | RSCANFDRF7*[1] | √ | √ | √ |
| DMACTRG[84] | Setting prohibited | — | — | — |
| DMACTRG[85] | INTDMAFL | √ | √ | √ |
| DMACTRG[86] | INTRLIN31UR0 | √ | √ | √ |
| DMACTRG[87] | INTRLIN31UR1 | √ | √ | √ |
| DMACTRG[88] | INTP7 | √ | √ | √ |
| DMACTRG[89] | INTCSIH2IC | √ | √ | √ |
| DMACTRG[90] | INTCSIH2IR | √ | √ | √ |
| DMACTRG[91] | INTCSIH2IJC | √ | √ | √ |
| DMACTRG[92] | INTTAUB0I1 | √ | √ | √ |
| DMACTRG[93] | INTTAUB0I3 | √ | √ | √ |
| DMACTRG[94] | INTTAUB0I5 | √ | √ | √ |
| DMACTRG[95] | INTTAUB0I7 | √ | √ | √ |
| DMACTRG[96] | INTTAUB0I8 | √ | √ | √ |
| DMACTRG[97] | INTTAUB0I10 | √ | √ | √ |
| DMACTRG[98] | INTTAUB0I12 | √ | √ | √ |
| DMACTRG[99] | INTTAUB0I14 | √ | √ | √ |
| DMACTRG[100] | INTTAUJ1I1 | √ | √ | √ |
| DMACTRG[101] | INTTAUJ1I3 | √ | √ | √ |
| DMACTRG[102] | INTP9 | — | √ | √ |
| DMACTRG[103] | INTADCA1I0 | — | √ | √ |
| DMACTRG[104] | INTADCA1I1 | — | √ | √ |
| DMACTRG[105] | INTADCA1I2 | — | √ | √ |
| DMACTRG[106] | ADC_CONV_END1 | — | √ | √ |
| DMACTRG[107] | Setting prohibited | — | — | — |
| DMACTRG[108] | Setting prohibited | — | — | — |
| DMACTRG[109] | INTCSIG1IC | — | √ | √ |
| DMACTRG[110] | INTCSIG1IR | — | √ | √ |
| DMACTRG[111] | INTRLIN33UR0 | √ | √ | √ |
| DMACTRG[112] | INTRLIN33UR1 | √ | √ | √ |
| DMACTRG[113] | INTRLIN35UR0 | — | √ | √ |
| DMACTRG[114] | INTRLIN35UR1 | — | √ | √ |
| DMACTRG[115] | INTTAUB1I1 | — | — | √ |
| DMACTRG[116] | INTTAUB1I3 | — | — | √ |
| DMACTRG[117] | INTTAUB1I5 | — | — | √ |
| DMACTRG[118] | INTTAUB1I7 | — | — | √ |
| DMACTRG[119] | INTTAUB1I8 | — | — | √ |
| DMACTRG[120] | INTTAUB1I10 | — | — | √ |
| DMACTRG[121] | INTTAUB1I12 | — | — | √ |
| DMACTRG[122] | INTTAUB1I14 | — | — | √ |
| DMACTRG[123] | Setting prohibited | — | — | — |
| DMACTRG[124] | Setting prohibited | — | — | — |

Table 8.6 DMA Trigger Factors (4/4)

| DMA Trigger Number DTFRm.REQSEL[6:0] | DMA Trigger Factor | F1K 100 pins | F1K 144 pins | F1K 176 pins |
|---|---|---|---|---|
| DMACTRG[125] | Setting prohibited | — | — | — |
| DMACTRG[126] | Setting prohibited | — | — | — |
| DMACTRG[127] | Setting prohibited | — | — | — |

Note 1. Supported only in PREMIUM products. Setting is prohibited in ECO/ADVANCED products.

## 8.2 Overview

### 8.2.1 Overview

Direct memory access (DMA) is used to access data without intervention of the CPU.

DMA includes a DMA transfer module called DMAC. A DMAC includes registers for storing transfer information. DMA has two 8-channel DMAC modules.

In this manual, DTFR denotes the function to select among hardware DMA transfer sources for a DMAC and retain the DMA transfer request. The DTFR can handle 128 types of hardware DMA transfer sources.

The address space that can be used for DMA transfer is a 4 GB address space represented by a 32-bit address. For information about which resource is assigned to a particular area in the 4 GB address space and which area is accessible from DMA, see **Section 4, Address Space**.

**CAUTION**

**DMA can be used after PEG access permission setting.**

**When accessing from the DMA to the local RAM and the retention RAM in the PE, the PEGSP register, PEGGnMK register (n = 0 to 3), PEGGnBA register (n = 0 to 3) and DMniCM register (ni = 00 to 07, 10 to 17) must be set.**

### 8.2.2 Term Definition

**Table 8.7** shows the terms used in this section.

Table 8.7 List of Term Definitions

| Term | Description |
|---|---|
| DMA transfer | A general term for data transfer carried out by DMA. |
| DMA cycle | A series of actions that consist of reading an amount of data specified by the transfer size (8/16/32/64/128 bits) from the address specified by the source address and writing it to the address specified by the destination address. The first half of the DMA cycle (reading part) is called a read cycle, and the second half (writing part) is called a write cycle. |
| Hardware DMA transfer source | A trigger for a DMA transfer request sent by an internal peripheral device. |
| Hardware DMA transfer request | A DMA transfer request generated by a hardware DMA transfer source |
| Software DMA transfer request | A DMA transfer request generated by software writing to a register. |
| DMA transfer request | A trigger to start DMA transfer with the DMAC. |
| Transfer information | The information required for DMA transfer, including the source address, destination address, transfer data size, and transfer count. |
| Single transfer | DMA transfer consisting of one DMA cycle started by one DMA transfer request. |
| Block transfer 1 | DMA transfer consisting of the number of DMA cycles specified by the transfer count in the transfer information, started by one DMA transfer request. |
| Block transfer 2 | DMA transfer consisting of the number of DMA cycles specified by the address reload count in the transfer information, started by one DMA transfer request. |
| Block transfer | A general term for both block transfer 1 and block transfer 2. |
| Last transfer | The DMA cycle carried out when the transfer count in the transfer information is 1. |
| Address reload transfer | The DMA cycle carried out when the address reload count in the transfer information is 1 if the reload function 2 is used. |
| Suspension | An action of pausing DMA transfer during block transfer. You can resume DMA transfer after suspension. |
| Resume | An action of resuming suspended DMA transfer. |
| Transfer abort | An action of aborting DMA transfer in the middle. You cannot resume DMA transfer after that. |

# 8.3 DMA Function

## 8.3.1 Basic Operation of DMA Transfer

### 8.3.1.1 Transfer Mode

DMA has three transfer modes.

#### Single Transfer

One DMA cycle is executed when a DMA transfer request is acknowledged.

#### Block Transfer 1

The number of DMA cycles specified in the transfer count register are executed when a DMA transfer request is acknowledged.

#### Block Transfer 2

The number of DMA cycles specified by the address reload count are executed when a DMA transfer request is acknowledged. If the value specified in the address reload count is larger than the value in the transfer count register, the number of DMA cycles specified in the transfer count register are executed.

### 8.3.1.2 Executing a DMA Cycle

DMA always executes a write cycle after a read cycle is complete.

For example, if the transfer data size is 128 bits, a write cycle is executed after a read cycle for the 128-bit data is finished. A write cycle never starts in the middle of a read cycle.

### 8.3.1.3 Updating Transfer Information

When a DMA cycle is executed, DMA updates transfer information as follows:

#### Source Address and Destination Address

Transfer information will be updated as described in **Table 8.8** according to the settings in the transfer control register such as the count directions of source address and destination address and transfer data size.

**Table 8.8 Updating the Source Addresses and the Destination Addresses**

| Count Direction | Transfer Data Size | Address after Update |
|---|---|---|
| Increment | 8 bits | (address before update) + $0000_0001_H$ |
| | 16 bits | (address before update) + $0000_0002_H$ |
| | 32 bits | (address before update) + $0000_0004_H$ |
| | 64 bits | (address before update) + $0000_0008_H$ |
| | 128 bits | (address before update) + $0000_0010_H$ |
| Decrement | 8 bits | (address before update) - $0000_0001_H$ |
| | 16 bits | (address before update) - $0000_0002_H$ |
| | 32 bits | (address before update) - $0000_0004_H$ |
| | 64 bits | (address before update) - $0000_0008_H$ |
| | 128 bits | (address before update) - $0000_0010_H$ |
| Fixed | — | Same as the address before update. |

When you use the reload function, a special update rule is applied other than the one described in **Table 8.8** for the last transfer and the address reload transfer. For details, see **Section 8.3.3, Reload Function**.

#### Transfer Count/Address Reload Count

The transfer count is decremented by one for every DMA cycle.

The address reload count is decremented by one for every DMA cycle when the reload function 2 or block transfer 2 is used. When the reload function 2 or block transfer 2 is not used, it is not updated.

When you use the reload function, a special update rule is applied for the last transfer and the address reload transfer. For details, see **Section 8.3.3, Reload Function**.

#### Other transfer information

Other transfer information is not updated during execution of a DMA cycle.

### 8.3.1.4 Last Transfer and Address Reload Transfer

The last transfer means a DMA cycle executed when the value in the transfer count register, which shows the remaining number of transfers, is one. The last transfer differs in operation compared to other DMA cycles as follows.

- The transfer completion flag (DCSTm.TC) is set when the last transfer is complete.
  The channel operation enable (DCENm.DTE) bit is cleared when the last transfer is complete (when continuous transfer is disabled).
- When the transfer completion interrupt output enable is set, a transfer completion interrupt is output when the last transfer is complete.
- When the reload function 1 is enabled, the reload function 1 is executed at the timing of the last transfer. For details, see **Section 8.3.3, Reload Function**.

The address reload transfer means a DMA cycle executed when the reload function 2 is enabled and the address reload count is one. The reload function 2 is executed during the address reload transfer. For details, see **Section 8.3.3, Reload Function**.

### 8.3.1.5 Transfer Completion Interrupt Output

DMA can output a transfer completion interrupt to external devices.

#### Transfer Completion Interrupt Output

When the transfer completion interrupt output enable (DTCTm.TCE) is set in the transfer control register, a DMAC requests a transfer completion interrupt when the last transfer is complete.

**Figure 8.1** shows the operation of the transfer completion interrupt.

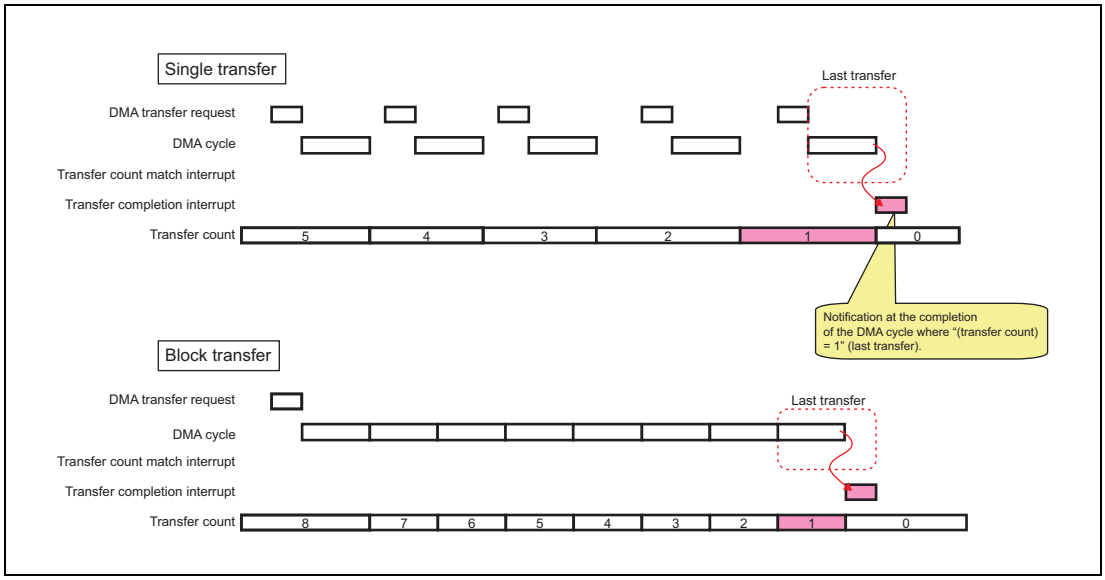



**Figure 8.1 Transfer Completion Interrupt**

### 8.3.1.6 Continuous Transfer

If the continuous transfer is not used, a DMAC sets the transfer completion flag (DCSTm.TC) and clears the channel operation enable (DCENm.DTE) bit when the last transfer is complete. In this case, a DMA transfer request is not accepted when the request is generated after the completion of the last transfer.

If the continuous transfer is used, the channel operation enable (DCENm.DTE) bit is not cleared when the last transfer is complete, and a DMA transfer request can be accepted even when the transfer completion flag is set. If DMA is used for a case where a specified number of DMA transfers are executed repetitively, software overhead associated with clearing the transfer completion flag and setting the channel operation enable bit after the completion of the last transfer can be reduced by using the continuous transfer.

The continuous transfer is enabled by setting the continuous transfer enable (DTCTm.MLE) in the DMAC transfer control register.

The continuous transfer is designed to work with the reload function 1. The continuous transfer function itself does not update the source address register, destination address register, and transfer count register. If, after the last transfer is complete, you want to restore the source address register, destination address register, and transfer count register to the state before the DMA transfer starts, use the reload function 1 and set the values of the source address register, destination address register, and the transfer count register before the DMA transfer starts to the reload source address register, reload destination address register, and reload transfer count register respectively.

**Figure 8.2** shows an operation of continuous transfer by a DMAC.

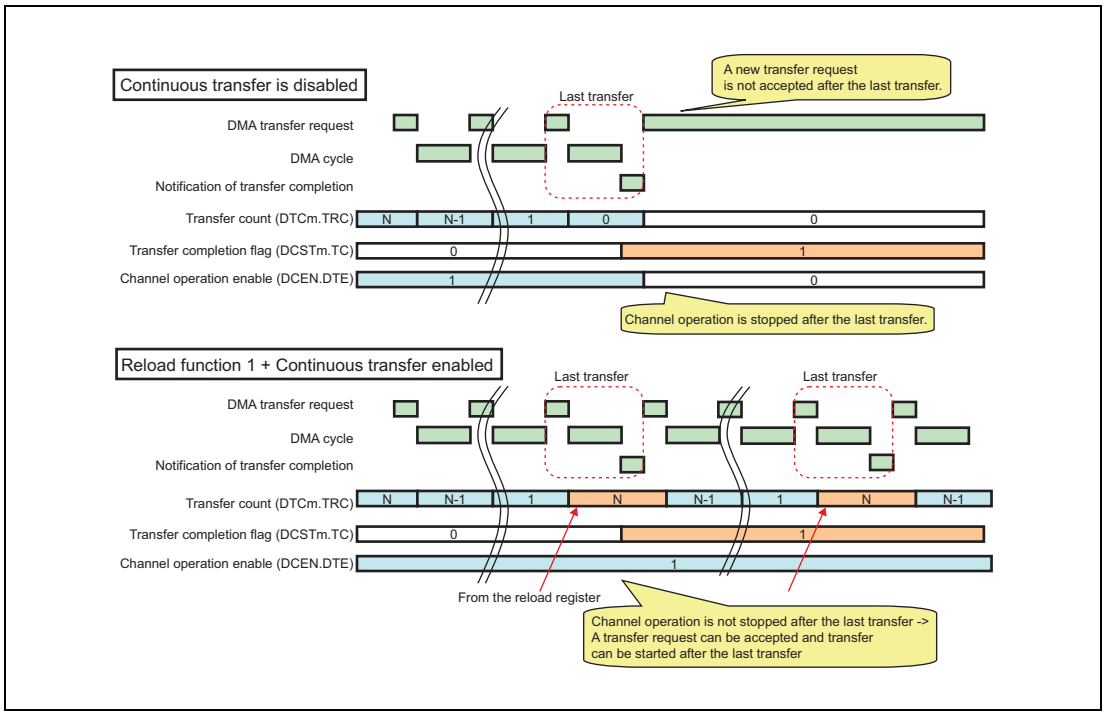



**Figure 8.2 Operation of Continuous Transfer by a DMAC**

## 8.3.2 Channel Priority Order

This subsection explains arbitration between multiple DMA channels.

### 8.3.2.1 DMAC Channel Arbitration

A DMAC select one channel out of eight channels with arbitration. Arbitration is done according to the fixed priority order. The priority order is “channel 0 > channel 1 > channel 2 > channel 3 > channel 4 > channel 5 > channel 6 > channel 7” for DMAC0, and “channel 8 > channel 9 > channel 10 > channel 11 > channel 12 > channel 13 > channel 14 > channel 15” for DMAC1.

Arbitration is done for every DMA cycle. No arbitration occurs between the read and write of a DMA cycle.

If, at the timing when one DMA cycle completes in the middle of a block transfer of a channel, there is a DMA transfer request from a channel with a higher priority, a DMA cycle of the channel with the higher priority will be executed as the result of arbitration.

If a DMAC executes the block transfer 1 or block transfer 2, DMAC channel arbitration is done for every DMA cycle, and possibly a DMA cycle of another DMAC channel with a higher priority may interrupt.

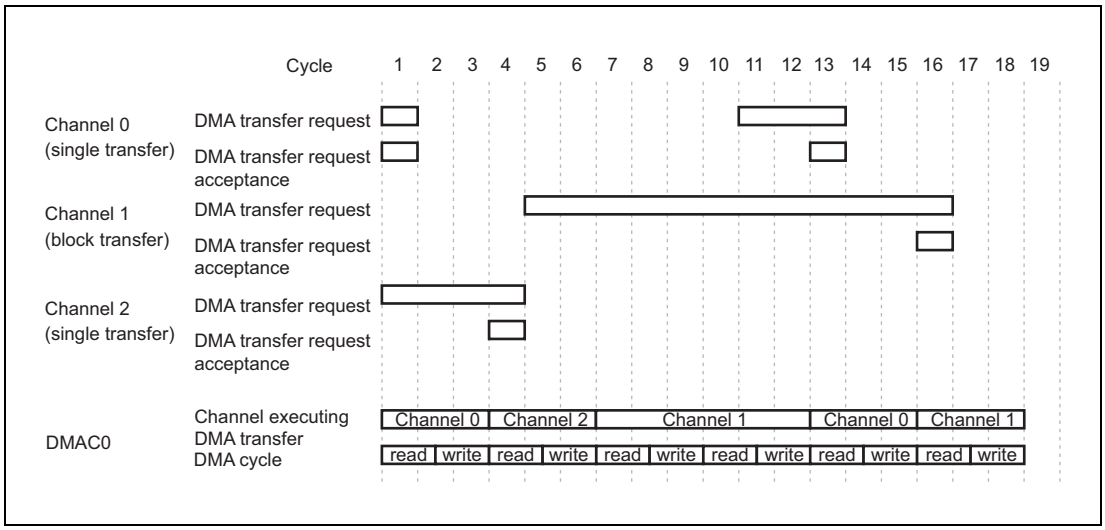



**Figure 8.3 DMAC Channel Arbitration**

Cycle numbers shown in **Figure 8.3** are for explanation purpose only. They do not indicate an actual number of cycles necessary for executing DMA transfer.

In **Figure 8.3**, DMA transfer requests for channels 0 and 2 are generated at Cycle 1. As a result of arbitration, a DMA cycle for channel 0 starts because its priority is higher. At Cycle 4, a DMA cycle for channel 2 starts. At Cycle 5, a DMA transfer request for channel 1 is generated. However, since the DMA cycle for channel 2 is still ongoing, no arbitration is done at this point. At Cycle 7, a DMA cycle for channel 1 starts. Because channel 1 uses block transfer, this DMA cycle continues at Cycle 10 where there are no other DMA transfer requests from other channel. At Cycle 11, a DMA transfer request for channel 0 is generated. However, since the DMA cycle for channel 1 is still ongoing, no arbitration is done at this point. At Cycle 12, the DMA cycle for channel 1 is complete. At Cycle 13, a DMA cycle for channel 0 starts as a result of arbitration between DMA channels 0 and 1.

It should be noted that, even though a block transfer of channel 1 has been already started, a DMA cycle of not channel 1 but channel 0 is executed at Cycle 13 because the priority of the latter is higher. At Cycle 15, the DMA cycle for channel 0 is complete. At Cycle 16, a DMA cycle for channel 1 starts again. At Cycle 18, the last DMA cycle of the block transfer of channel 1 is complete.

#### 8.3.2.2 Interface Arbitration

DMAC0 and DMAC1 work independently and execute DMA transfer.

## 8.3.3 Reload Function

### 8.3.3.1 Overview of the Reload Function

The reload function updates a portion of transfer information, more specifically, the source address, destination address, transfer count, and address reload count, to the predefined values during DMA transfer.

The reload function has two types of functions: reload function 1 and reload function 2.

### 8.3.3.2 Operation of Reload Function 1

When the reload function 1 is enabled, actions described in **Table 8.9** are executed at the timing of the last transfer according to the reload function 1 setting.

**Table 8.9 Operation of Reload Function 1**

| Reload Function 1 Setting | Register | Action at the Last Transfer |
|---|---|---|
| 00<br>(Reload function 1 disabled.) | Source address | Not reloaded. |
| | Destination address | Not reloaded. |
| | Transfer count | Not reloaded. |
| | Address reload count | Not reloaded. |
| 01<br>(Reload function 1 enabled. Reloading source address and transfer count.) | Source address | The reload source address is copied to this. |
| | Destination address | Not reloaded. |
| | Transfer count | The reload transfer count is copied to this. |
| | Address reload count | • If the reload function 2 is disable: Not reloaded.<br>• If the reload function 2 is enabled: The reload address reload count is copied to this. |
| 10<br>(Reload function 1 enabled. Reloading destination address and transfer count.) | Source address | Not reloaded. |
| | Destination address | The reload destination address is copied to this. |
| | Transfer count | The reload transfer count is copied to this. |
| | Address reload count | • If the reload function 2 is disable: Not reloaded.<br>• If the reload function 2 is enabled: The reload address reload count is copied to this. |
| 11<br>(Reload function 1 enabled. Reloading source address, destination address, and transfer count.) | Source address | The reload source address is copied to this. |
| | Destination address | The reload destination address is copied to this. |
| | Transfer count | The reload transfer count is copied to this. |
| | Address reload count | • If the reload function 2 is disable: Not reloaded<br>• If the reload function 2 is enabled: The reload address reload count is copied to this. |

**Figure 8.4** shows an operation of the reload function 1.

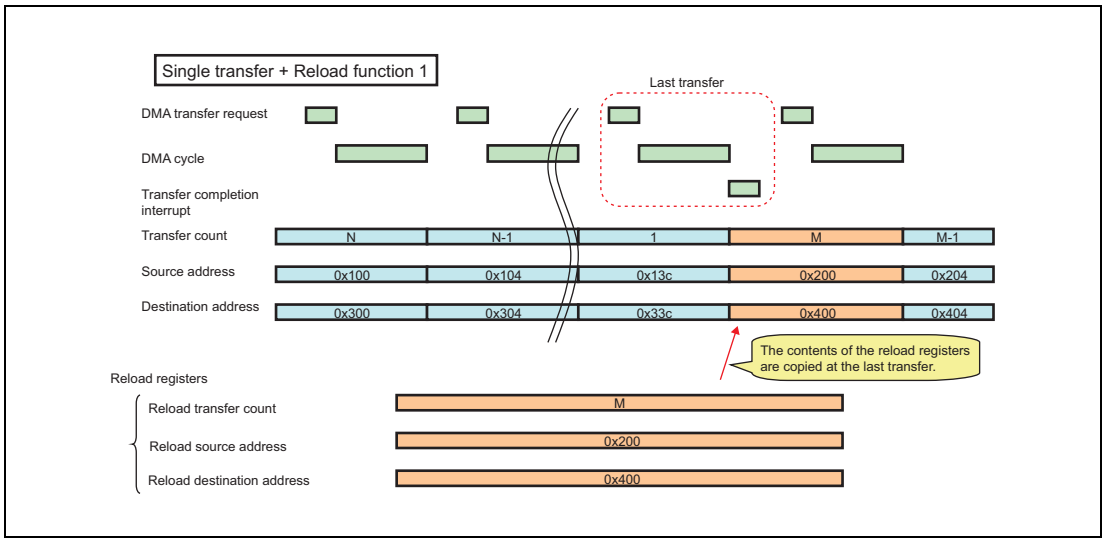



**Figure 8.4 Operation of Reload Function 1**

### 8.3.3.3 Reload Function 2

When the reload function 2 is enabled, actions described in **Table 8.10** are executed at the timing of the address reload transfer according to the reload function 2 setting.

**Table 8.10 Operation of Reload Function 2**

| Reload Function 2 Setting | Register | Action at the Address Reload Transfer |
|---|---|---|
| 00<br>(Reload function 2 disabled.) | Source address | Not reloaded. |
| | Destination address | Not reloaded. |
| | Address reload count | Not reloaded. |
| 01<br>(Reload function 2 enabled.<br>Reloading source address and address reload count.) | Source address | The reload source address is copied to this. |
| | Destination address | Not reloaded. |
| | Address reload count | The reload address reload count is copied to this. |
| 10<br>(Reload function 2 enabled.<br>Reloading destination address and address reload count.) | Source address | Not reloaded. |
| | Destination address | The reload destination address is copied to this. |
| | Address reload count | The reload address reload count is copied to this. |
| 11<br>(Reload function 2 enabled.<br>Reloading source address and destination address and address reload count.) | Source address | The reload source address is copied to this. |
| | Destination address | The reload destination address is copied to this. |
| | Address reload count | The reload address reload count is copied to this. |

**Figure 8.5** shows an operation of the reload function 2.

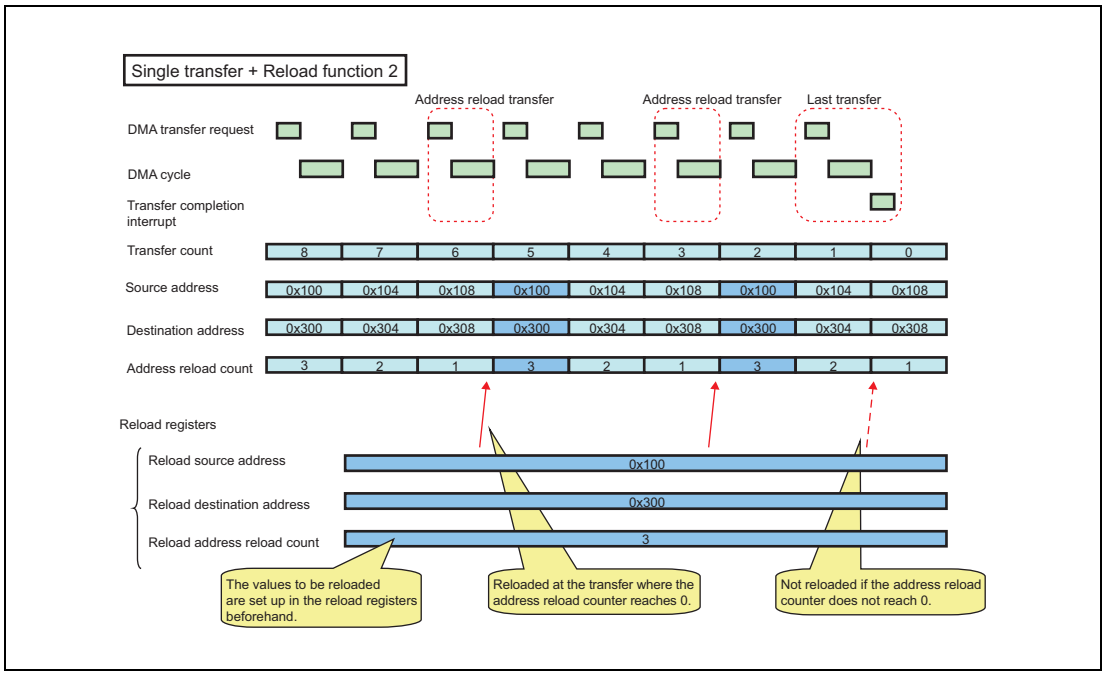



**Figure 8.5 Operation of Reload Function 2**

**Figure 8.6** shows an operation when both the reload function 1 and the reload function 2 are used simultaneously.

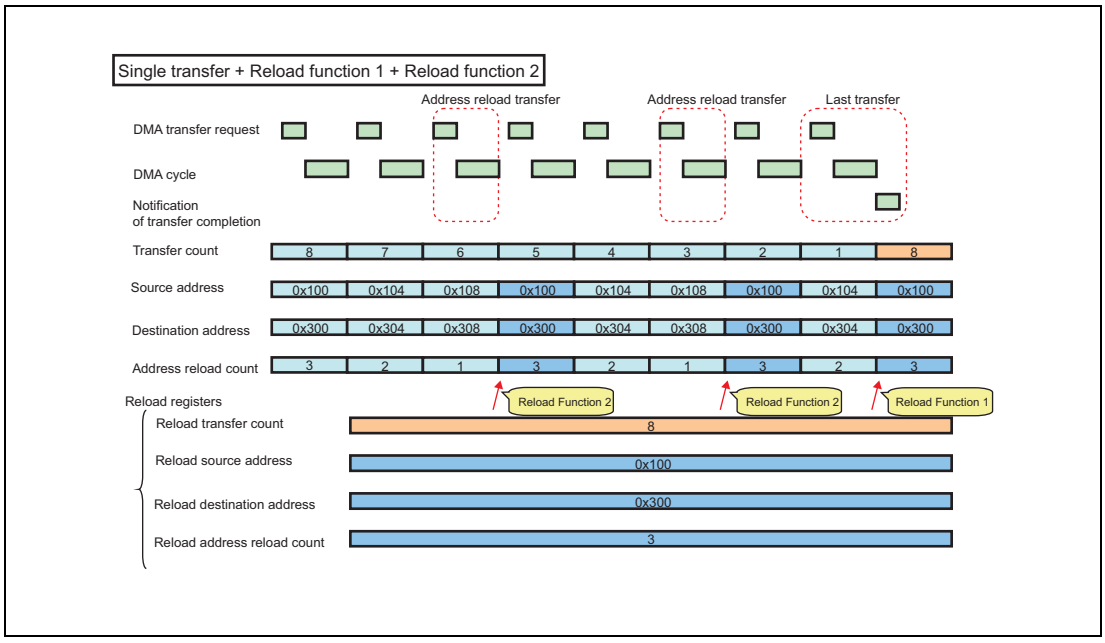



**Figure 8.6 Operation when Combining the Reload Function 1 and the Reload Function 2**

#### 8.3.3.4 Timing of Setting DMAC Reload Registers

You can set up the reload source address register, reload destination address register, and reload transfer count register any time (even during DMA transfer). However, if you update the reload source address register, reload destination address register, and reload transfer count register during DMA transfer, there may be a conflict between reloading at the last transfer or address reload transfer and update of the reload register by users. In order to avoid this conflict, setting up reload registers must be completed before the last transfer or address reload transfer starts.

## 8.3.4 Chain Function

### 8.3.4.1 Overview

DMA offers a function called a chain function. With this function, the completion of the DMA cycle or last transfer for one channel can trigger a DMA transfer request for another channel. A DMA transfer request for another channel initiated by the chain function is called a chain request.

You can select the condition for generating a chain request from the following two options:

- Always chain: A chain request is generated at the completion of every DMA cycle.
- Chain at the last transfer: A chain request is generated at the completion of the last transfer.

**Figure 8.7** shows the operation of "always chain."

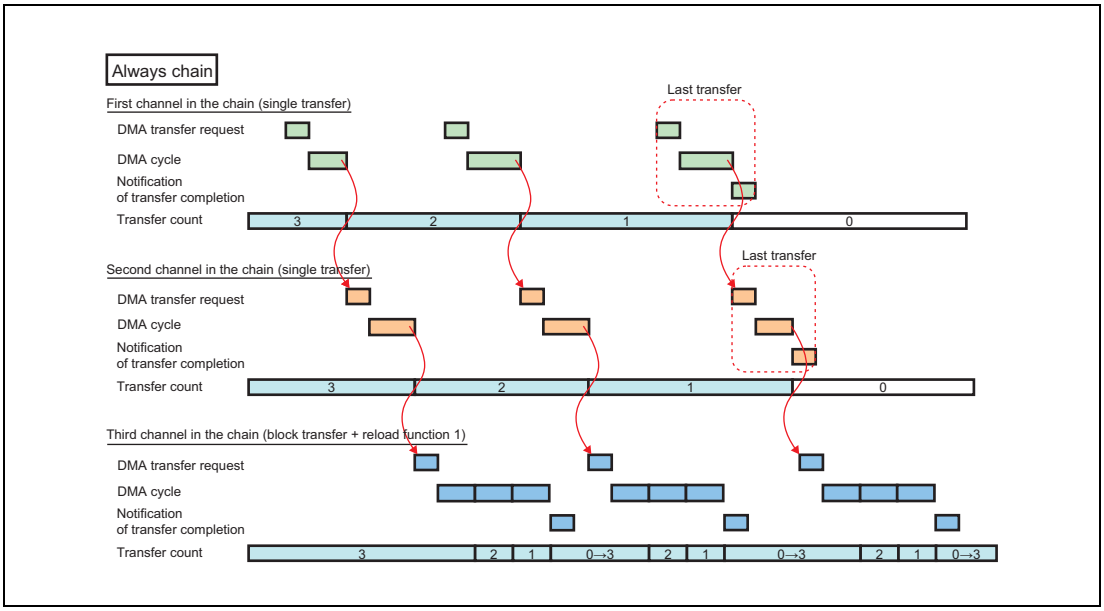



Figure 8.7 Operation of "Always Chain"

**Figure 8.8** shows the operation of "chain at the last transfer."

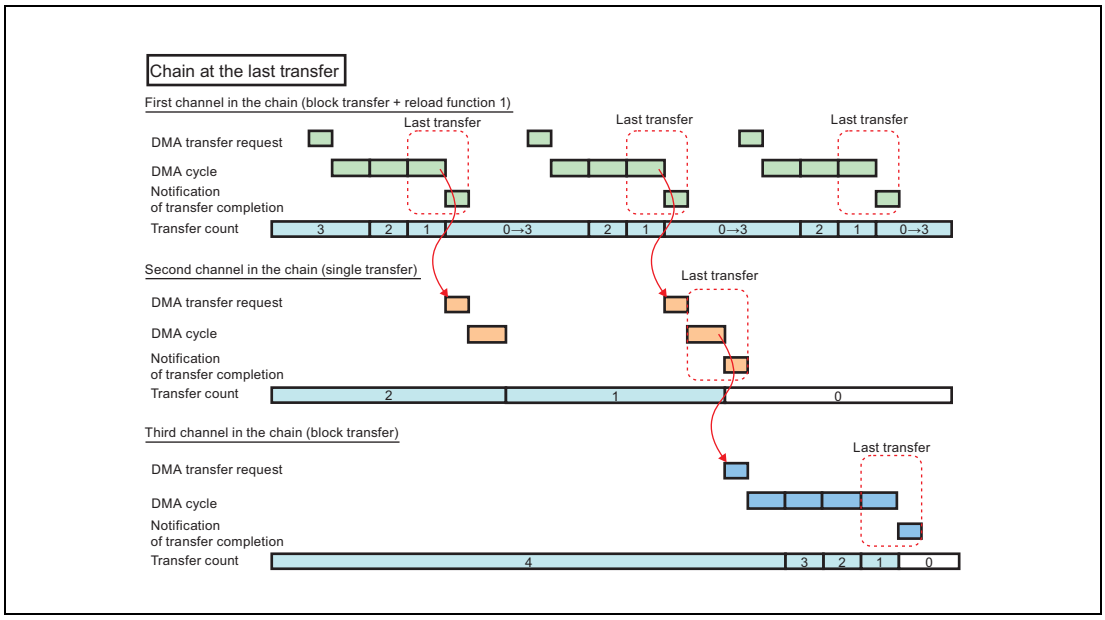



**Figure 8.8 Operation of "Chain at the Last Transfer"**

### 8.3.4.2 Setting Up the Chain Function

For a DMAC, you need to write to the chain enable (DTCTm.CHNE) and the next channel in the chain selection (DTCTm.CHNSEL) in the DMAC transfer control register in order to set up the type of chain function and the next channel number in the chain.

### 8.3.4.3 Caution for Using the Chain Function

The chain function is enabled by setting the software DMA transfer request flag of the next channel in the chain as a part of its function. Therefore, you need to set up the channel settings of the next channel in the chain in the same way as when the software DMA transfer request is used. If you specify a channel using the hardware DMA transfer request for the next channel in the chain, the chain function does not work.

A channel and its next channel in the chain must belong to the same module (DMAC0 and DMAC1). You cannot specify a channel in another module for its next channel in the chain.

## 8.3.5 DMAC Operation

### 8.3.5.1 Types of DMA Transfer Requests and Assigning DMA Transfer Requests

A DMAC starts DMA transfer by accepting a hardware DMA transfer request or software DMA transfer request. The DMA transfer request selection assignment (DTCTm.DRS) bit in the DMAC transfer control register (DTCTm) determines whether a hardware DMA transfer request or a software DMA transfer request is used.

In the case of a hardware DMA transfer request for a DMAC, DTFR selects one out of 128 hardware DMA transfer factor and assigned to each channel of the DMAC. This assignment is configured in the DTFR setting registers.

### 8.3.5.2 Generating and Accepting a Software DMA Transfer Request

By setting the software DMA transfer request flag (DCSTm.SR) in the DMAC transfer status register (DCSTm) using the DMAC transfer status set register (DCSTSm), a software DMA transfer request can be generated.

The software DMA transfer request flag is automatically cleared when the DMAC processes the DMA transfer request. The timing when the software DMA transfer request flag is automatically cleared differs depending on the transfer mode of the DMA transfer to be executed.

- In the single transfer mode, the software DMA transfer request flag is cleared whenever the software DMA transfer request is accepted.
- In the block transfer 1 mode, the software DMA transfer request flag is cleared when the last transfer starts.
- In the block transfer 2 mode, the software DMA transfer request flag is cleared when the last transfer or address reload transfer starts.

The software DMA transfer request flag can also be cleared by software using the DMAC transfer status clear register (DCSTCm). When you abort a DMA transfer of a DMAC channel, you must clear the software DMA transfer request flag.

### 8.3.5.3 Executing DMA Transfer

When the DMAC accepts a DMA transfer request for a channel, the DMAC executes DMA transfer of the channel. If there are DMA transfer requests from multiple channels, DMAC does DMAC channel arbitration and decides a channel to be acknowledged.

## 8.4 Suspension, Resume, Transfer Abort, and Clearing of a DMA Transfer Request

### 8.4.1 DMA Suspension and Resume by Software Control

The DMA control register (DMACTL) is used to suspend DMA transfer for all channels.

When the DMA suspension bit (DMACTL.DMASPD) in the DMA control register is set, DMA puts all channels into the suspended state. If all channels are in the suspended state and the DMA suspension bit in the DMA control register is cleared, DMA restores all channels from the suspended state to the normal state and resumes the DMA transfer of the suspended channel.

When all channels are in the suspended state, DMA transfer is suspended for all channels without changing the value of the DCENm.DTE bit of each DMAC channel.

### 8.4.2 Suspension, Resume, and Transfer Abort of a DMAC Channel

You can suspend the DMA transfer of a DMAC channel by clearing the channel operation enable bit (DCENm.DTE) in the DMAC channel operation enable setting register for the channel. If a DMA cycle is ongoing, the DMA transfer of the channel is suspended after the currently ongoing DMA cycle is finished. If you set the DCENm.DTE bit again while the DMA transfer of the channel is suspended, the DMA transfer of the channel is resumed.

If you want to abort the currently ongoing DMA transfer of a DMAC channel, similarly clear the channel operation enable bit (DCENm.DTE) in the DMAC channel operation enable setting register, and then clear the hardware DMA transfer request in the DTFR in the case of a hardware DMA transfer request, or clear the software DMA transfer request flag (DCSTm.SR) using the DMAC transfer request flag clear bit (DCSTCm.SRC) in the DMAC transfer status clear register in the case of a software DMA transfer request.

In case that the continuous transfer enable bit (DTCTm.MLE) is set, the channel operation enable bit (DCENm.DTE) is kept to be set. Even though the channel operation enable bit (DCENm.DTE) is cleared by software during a DMA cycle in a last transfer, the function of the continuous transfer enable bit (DTCTm.MLE) is given high priority and the channel operation enable bit (DCENm.DTE) is set after completion of the last transfer.

If you want to abort an ongoing DMA transfer of a DMAC channel when continuous transfer function is enabled, please clear the continuous transfer enable bit (DTCTm.MLE) first and then clear the channel operation enable bit (DCENm.DTE) to abort DMA transfer of the DMAC channel. Only for the operation, DMAC Transfer Control Register (DTCTm) can be written under the channel operation is enabled (DCENm.DTE = 1).

**Figure 8.9** shows an example of suspension, resume, and transfer abort of a DMAC channel.

In **Figure 8.9**, both channels 0 and 1 execute block transfer. At time tick 1, DMA transfer of channel 1 starts. At time tick 2, a DMA transfer request for channel 0 is accepted. As a result of DMAC channel arbitration, DMA transfer of channel 0 starts because channel 0 has a higher priority than channel 1. At time tick 3, the last transfer of channel 0 is complete, and the remaining DMA transfer in the block transfer of channel 1 starts. At time tick 4, the last transfer of channel 1 is complete. After time tick 5, DMA transfer of channel 0 and DMA transfer of channel 1 are executed similarly. At time tick 7, the DMA transfer of channel 0 is suspended and, as a result of DMAC channel arbitration, the DMA transfer of channel 1 starts. At time tick 8, the last transfer of channel 1 is complete, and then, at time tick 9, the DMA transfer of channel 0 resumes. At time tick 10, the DMA transfer of channel 0 is suspended again, and then, at time tick 11, the DMA transfer of channel 0 is aborted. At time tick 12,

the suspended state for channel 0 is cleared, but the DMA transfer of channel 0 is not executed because the DMA transfer is aborted at time tick 11.

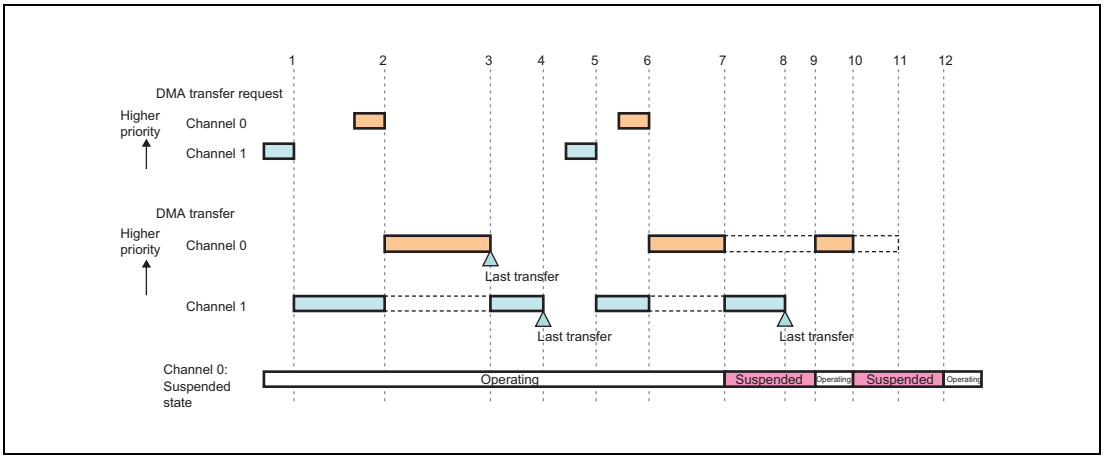



**Figure 8.9 Example of Suspension, Resume, and Transfer Abort of a DMAC Channel**

## 8.4.3 Masking and Clearing a Hardware DMA Transfer Request by the DTFR

If a DMAC uses a hardware DMA transfer request, you can temporarily disable (mask) the hardware DMA transfer request output from the DTFR to the DMAC by clearing the hardware DMA transfer source selection enable bit (DTFRm.REQEN) in the DTFR setting register.

Also, if a hardware DMA transfer source is used, you can clear a hardware DMA transfer request retained in the DTFR by using the hardware DMA transfer request clear (DTFRRQCm.DRQC) bit in the DTFR transfer request clear register.

Even if you suspend or abort DMA transfer of a DMAC channel, the hardware DMA transfer request selection/hold circuit of the DTFR is still running, and consequently, the DTFR may retain a hardware DMA transfer request that was input to the DTFR during the suspension or transfer abort period of the DMAC channel. When you resume or start DMA transfer of a DMAC channel, clear the hardware DMA transfer request retained in the DTFR as required.

In case that DMAC is used both with hardware DMA transfer request and with block transfer (1 or 2) mode, if DTFR hardware DMA transfer source selection enable bit is set to disable (DTFRn.REQEN = 0) by software while DMAC is executing block transfer, the ongoing block transfer is suspended.

### 8.4.4 List of Suspend, Resume, and Transfer Abort Functions

Table 8.11 List of Suspend, Resume, and Transfer Abort Functions

| Function | How to execute the function | Operation | Possibility of DMA transfer abort | Master that can execute the function (See Section 8.6, Reliability Functions.) |
|---|---|---|---|---|
| DMA suspension and resume by software control | Setting and clearing the DMACTL.DMASPD | All channels are in the suspended state. | Not possible*1 | Special master |
| Suspension and resume of a DMAC channel | Clearing and setting the DCENm.DTE in each channel register*2 | DMA transfer of a channel is suspended. | Possible (by clearing the DMA transfer request flag in suspension state) | Special master, and general master assigned to the channel. |

Note 1. In order to abort DMA transfer, you need to abort transfer for the DMAC channel.

Note 2. In case that the continuous transfer enable bit (DTCTm.MLE) is set, please clear the continuous transfer enable bit (DTCTm.MLE) first before clearing the DCENm.DTE bit to suspend the channel and set the continuous transfer enable bit (DTCTm.MLE) before setting the DCENm.DTE bit to resume the channel.

## 8.5 Error Control

### 8.5.1 Type of Error

DMA can generate the following type of error.

- DMA Transfer Error
  This error is generated when error is detected in the read cycle or write cycle in a DMA cycle.
  This error can be generated in all DMAC channels during execution of DMA transfer.
  A DMA transfer error leads to INTDMAERR interrupt request which is a source of FEINT.

### 8.5.2 DMA Transfer Error

#### 8.5.2.1 Operation of a DMAC When DMA Transfer Error Occurs

When DMA transfer error occurs in a DMAC, the transfer error flag (DCSTm.ER) in the DMAC transfer status register of the channel where the DMA transfer error occurred is set. The DMAC error register (DMACER) shows the transfer error flags of all 16 DMAC channels.

On a channel where the transfer error flag is set, a new DMA cycle is not executed if the transfer disable on transfer error setting (DTCTm.ESE) bit is set. On the other hand, a DMA cycle is executed regardless of the value of the transfer error flag if the transfer error case DMA transfer disable setting (DTCTm.ESE) bit is cleared.

If you want to abort the DMA transfer of a channel where the DMA transfer error occurred is set, follow the procedure to abort DMA transfer of the DMAC channel.

If DMA transfer error occurs during the read cycle of a DMA cycle, the write cycle is not executed. If DMA transfer error occurs during the write cycle of a DMA cycle, the validity of the result of write operation is not guaranteed.

Regardless of whether DMA transfer error occurs in the read cycle or write cycle of a DMA cycle, the source address register, destination address register, transfer count register, and address reload count register are updated.

## 8.6 Reliability Functions

### 8.6.1 Overview

In this product, DMA provides the following reliability functions:

- Register access protection function
- Master information inherit function

### 8.6.2 Register Access Protection Function

The register access protection function allows write access to the transfer information of each DMA channel only from the master assigned to the channel and prohibits write access from other masters.

The register access protection function enables you, for example, to prevent the settings of the channel from being updated by masters other than the one assigned to the channel.

#### 8.6.2.1 Identifying the Accessing Master

DMA identifies a master based on the processor element ID number (PEID) of the accessing master, the system protection ID configured by the accessing CPU (SPID), and whether the CPU is in the supervisor mode (PSW.UM=0) or the user mode (PSW.UM=1).

#### 8.6.2.2 Master Access

DMA handles accesses from CPU in the supervisor mode (PSW.UM = 0) as accesses from special masters. Special masters are allowed to read from or write to all DMA registers.

DMA handles the other masters except special masters as general masters. General masters are allowed to read from all DMA registers, but allowed to write to the following specific registers.

- Channel registers of the channels assigned by the channel assignment. (For details, see **Section 8.6.2.3, Channel Assignment**.)

From general masters, write access to registers other than the above is not allowed.

#### 8.6.2.3 Channel Assignment

To each channel, DMA can assign a master so that the master is allowed to use the channel. Channel assignment is configured in the channel master setting register (DMniCM) by the CPU in the supervisor mode (PSW.UM = 0).

In general master access, the master assigned to a channel by the channel assignment is allowed to write the channel registers of the channel. If the channel registers of a channel is written by a master other than the master assigned to the channel, the access is called illegal access. For information about illegal access, see **Section 8.6.2.4, Illegal Access**.

#### 8.6.2.4 Illegal Access

DMA handles the following access as illegal access.

(a) Write access to global registers from general masters

(b) Write access to channel registers from general masters who are not assigned to the channel

DMA never treat read access from any master as illegal access.

DMA's actions against illegal access are as follows:

For both cases (a) and (b),

- Write access is ignored.

Only for the case (b),

- The information about the illegal access is stored in a register access protection violation register.
- The DMAC0 and DMAC1 have their own register access protection violation registers (DM0CMV and DM1CMV respectively).

Only the special master can access the register access protection violation registers. The special master can check whether illegal access has occurred by checking the register access protection violation registers periodically.

In addition, it is recommended that, when a master uses DMA and configures transfer information in the channel registers, the master should check whether the configuration has been successfully completed without illegal access by, for example, reading back the settings.

### 8.6.3 Master Information Inherit Function

In this product, DMA inherits master information that is equivalent to the master information of the master to which the DMA channel is assigned.

The master information that is output from DMA is shown in **Table 8.12**.

**Table 8.12 Master Information That is Output from DMA**

| Meaning | Value that is output from DMA |
|---|---|
| UM | UM bit value in the channel master setting register |
| SPID | SPID bit value in the channel master setting register |
| PEID | PEID bit value in the channel master setting register |
| DMA | 1 |

### 8.6.4 Other Reliability Functions

#### 8.6.4.1 Restriction on the Next Channel in the Chain

The reliability function limits the channels you can select as the next channel in the chain.

When you use the chain function, the channel master settings of a channel and its next channel in the chain must be the same.

The chain function is designed so that a channel and its next channel in the chain are managed by the same master.

When DMA detects that different masters are assigned to a channel and its next channel in the chain, it is deemed illegal and the chain function is suppressed. More specifically, when DMA tries to execute the chain function, DMA compares the chain master settings of the channel and its next channel in the chain, and if the settings are the same for both PEID and UM, the chain function is allowed and a chain request is sent to the next channel. If the channel master settings are not the same for either PEID or UM, a chain request is not sent.

# 8.7 Setting Up DMA Transfer

## 8.7.1 Overview of Setting Up DMA

Table 8.13 Overview of Setting Up DMA

| No. | Master that Configures the Setting | Description | Register | | Necessity of the Setting |
|---|---|---|---|---|---|
| 1 | Special master (CPU in the supervisor mode (UM = 0)) | Overall DMA operation setting | DM00CM to DM17CM | DMAC channel master setting | Mandatory |
| 2 | | Status clear | CMVC | Channel protection violation clear register | Recommended |
| 3 | Master assigned to the DMAC channel | Channel setting | DSAm | DMAC source address | Mandatory |
| 4 | | | DDAm | DMAC destination address | Mandatory |
| 5 | | | DTCm | DMAC transfer count | Mandatory |
| 6 | | | DTCTm | DMAC transfer control | Mandatory |
| 7 | | | DRSAm | DMAC reload source address | Mandatory if the reload function is used |
| 8 | | | DRDAm | DMAC reload destination address | Mandatory if the reload function is used |
| 9 | | | DRTCm | DMAC reload transfer count | Mandatory if the reload function is used |
| 10 | | | DTFRm | DTFR setting register | Mandatory |
| 11 | | Status clear | DCSTCm | DMAC transfer status clear | Mandatory |
| 12 | | | DTFRRQCm | DTFR transfer request clear | Recommended |
| 13 | | Channel operation enable | DCENm | DMAC channel operation enable setting | Mandatory |

## 8.7.2 Setting Up the Overall DMA Operation

You need to set up the overall DMA operation before you start using DMA.

To configure the overall DMA operation, the special master (a CPU in the supervisor mode (UM = 0)) needs to set up global registers. Global registers can be set up only by special master access. For details, see **Section 8.6, Reliability Functions**.

The following register must be set up to configure the overall DMA operation.

- DMAC channel master setting registers (DMniCM)

These registers configure channel assignment. (For details, see **Section 8.6, Reliability Functions**.)

If the DMAC channel master setting registers are not properly set, DMA channel setting and DMA transfer cannot be executed properly.

Also, if errors are detected in the following registers while the overall DMA operation is set up, clearing the errors is recommended.

- DMAC0 register access protection violation register (DM0CMV)
- DMAC1 register access protection violation register (DM1CMV)

## 8.7.3 Setting Up the DMA Channel Setting

The DMA channel setting defines the transfer information and transfer source for each DMAC.

To configure the DMA channel setting, each channel's master assigned by the channel assignment sets channel registers.

### 8.7.3.1 Setting Up the DMAC Channel Setting

Follow the procedure below to set up the DMAC channel setting in case of using DMAC.

#### (1) Disabling the DMAC Channel Operation

If the channel operation enable (DCENm.DTE) in the DMAC channel operation enable setting register (DCENm) is set, clear the DCENm.DTE bit to disable the channel operation.

#### (2) Setting Up the Transfer Information

When you set up the transfer information of the DMAC, the following registers need to be set up.

- DMAC source address register (DSAm)
- DMAC destination address register (DDAm)
- DMAC transfer count register (DTCm)
- DMAC transfer control register (DTCTm)
- DMAC reload source address register (DRSAm)
- DMAC reload destination address register (DRDAm)
- DMAC reload transfer count register (DRTCm)

### (3) Setting Up the DMA Transfer Request

While setting the transfer information, you need to set up the DMA transfer request selection assignment (DTCTm.DRS) bit in the DMAC transfer control register (DTCTm) to define whether the hardware or software DMA transfer request is used.

You cannot use both the hardware and software DMA transfer requests for the same channel at the same time.

If you use the hardware DMA transfer request, you need to select the source used as the hardware DMA transfer request out of 128 hardware DMA transfer sources using the hardware DMA transfer source selection (DTFRm.REQSEL) in the DTFR setting register. Also, you need to enable the hardware DMA transfer source selection (DTFRm.REQEN) in the same register.

The DTFR may retain a hardware DMA transfer request before the hardware DMA transfer source is selected. Clear the hardware DMA transfer request (DTFRRQm.DRQ) retained in the DTFR using the DTFR transfer request clear register (DTFRRQCm) if necessary.

If you use the software DMA transfer request, disable the hardware DMA transfer source selection (DTFRm.REQEN) in the DTFR setting register.

### (4) Clearing the Transfer Status

The DMAC transfer status register (DCSTm) may retain the result of the previous DMA transfer, so clear the flags in the DMAC transfer status register using the DMAC transfer status clear register (DCSTCm).

### (5) Enabling the DMAC Channel Operation

Set the channel operation enable (DCENm.DTE) bit in the DMAC channel operation enable setting register to enable the channel operation.

After the channel operation enable bit is set, the DMAC can accept a DMA transfer request and become ready for DMA transfer.

# 8.8 Global Registers

## 8.8.1 List of Global Register Address

The global register addresses are listed in the table below.

For <DMA_base>, see **Section 8.1.2, Register Base Address**.

**Table 8.14 List of Global Register Addresses**

| Unit Name | Register Name | Register Symbol | Address | Access Permission | |
|---|---|---|---|---|---|
| | | | | Special Master | General Master |
| DMAC | DMA control register | DMACTL | <DMA_base> + $0000_H$ | √ | —*1 |
| | DMAC error register | DMACER | <DMA_base> + $0020_H$ | √ | √ |
| | DMAC0 register access protection violation register | DM0CMV | <DMA_base> + $0030_H$ | √ | —*1 |
| | DMAC1 register access protection violation register | DM1CMV | <DMA_base> + $0034_H$ | √ | —*1 |
| | Register access protection violation clear register | CMVC | <DMA_base> + $003C_H$ | √ | —*1 |
| | DMAC0 channel 0 channel master setting | DM00CM | <DMA_base> + $0100_H$ | √ | —*1 |
| | DMAC0 channel 1 channel master setting | DM01CM | <DMA_base> + $0104_H$ | √ | —*1 |
| | DMAC0 channel 2 channel master setting | DM02CM | <DMA_base> + $0108_H$ | √ | —*1 |
| | DMAC0 channel 3 channel master setting | DM03CM | <DMA_base> + $010C_H$ | √ | —*1 |
| | DMAC0 channel 4 channel master setting | DM04CM | <DMA_base> + $0110_H$ | √ | —*1 |
| | DMAC0 channel 5 channel master setting | DM05CM | <DMA_base> + $0114_H$ | √ | —*1 |
| | DMAC0 channel 6 channel master setting | DM06CM | <DMA_base> + $0118_H$ | √ | —*1 |
| | DMAC0 channel 7 channel master setting | DM07CM | <DMA_base> + $011C_H$ | √ | —*1 |
| | DMAC1 channel 0 channel master setting | DM10CM | <DMA_base> + $0120_H$ | √ | —*1 |
| | DMAC1 channel 1 channel master setting | DM11CM | <DMA_base> + $0124_H$ | √ | —*1 |
| | DMAC1 channel 2 channel master setting | DM12CM | <DMA_base> + $0128_H$ | √ | —*1 |
| | DMAC1 channel 3 channel master setting | DM13CM | <DMA_base> + $012C_H$ | √ | —*1 |
| | DMAC1 channel 4 channel master setting | DM14CM | <DMA_base> + $0130_H$ | √ | —*1 |
| | DMAC1 channel 5 channel master setting | DM15CM | <DMA_base> + $0134_H$ | √ | —*1 |
| | DMAC1 channel 6 channel master setting | DM16CM | <DMA_base> + $0138_H$ | √ | —*1 |
| | DMAC1 channel 7 channel master setting | DM17CM | <DMA_base> + $013C_H$ | √ | —*1 |

Note 1. The registers are read only for general master.

## 8.8.2 Details of Global Registers

### 8.8.2.1 DMACTL — DMA Control Register

**Access:** This register can be read or written in 32-bit units.

**Address:** <DMA_base> + $0000_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DMA SPD |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 8.15 DMACTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | DMASPD | DMA suspension<br>This bit indicates that DMA transfer for all channels is suspended. If a user writes 1 to this bit, DMA transfer for all channels can be suspended. If a user writes 0 to this bit, suspension of DMA transfer for all channels can be released. The suspension controlled by this bit is independent from the suspension controlled by the transfer enable bit (DCENm.DTE) of each DMAC channel. That means, if this bit is set to 1, all DMA transfers are suspended regardless of the values of the DCENm.DTE bit of each DMAC channel.<br>Writing to this bit does not affect the DCENm.DTE bit of each DMAC channel.<br>0: DMA suspension cleared<br>1: DMA suspension request/DMA suspension ongoing |

### 8.8.2.2 DMACER — DMAC Error Register

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <DMA_base> + $0020_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DM1 ER7 | DM1 ER6 | DM1 ER5 | DM1 ER4 | DM1 ER3 | DM1 ER2 | DM1 ER1 | DM1 ER0 | DM0 ER7 | DM0 ER6 | DM0 ER5 | DM0 ER4 | DM0 ER3 | DM0 ER2 | DM0 ER1 | DM0 ER0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.16 DMACER Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. |
| 15 to 8 | DM1ER[7:0] | DMAC1 DMA transfer error status<br>These bits show the DMA transfer error status of channels 0 through 7 of the DMAC1.<br>Each bit is mapped from the DCSTm.ER bit of each channel of the DMAC1 and is read-only.<br>0: DMA transfer error is not generated<br>1: DMA transfer error is generated |
| 7 to 0 | DM0ER[7:0] | DMAC0 DMA transfer error status<br>These bits show the DMA transfer error status of channels 0 through 7 of the DMAC0.<br>Each bit is mapped from the DCSTm.ER bit of each channel of the DMAC0 and is read-only.<br>0: DMA transfer error is not generated<br>1: DMA transfer error is generated |

### 8.8.2.3 DM0CMV — DMAC0 Register Access Protection Violation Register

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <DMA_base> + $0030_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | PEID2 | PEID1 | PEID0 | SPID1 | SPID0 | UM | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | VCH2 | VCH1 | VCH0 | — | — | — | VF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.17 DM0CMV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | When read, the value after reset is returned. |
| 22 to 17 | PEID[2:0]<br>SPID[1:0]<br>UM | Illegal access master information<br>These bits retain the accessing master information of the first illegal access after the DM0CMV.VF bit is cleared to 0.<br>If illegal access occurs while the DM0CMV.VF bit is 1, these bits do not change.<br>These bits are read-only and cannot be cleared. |
| 16 to 7 | Reserved | When read, the value after reset is returned. |
| 6 to 4 | VCH[2:0] | Channel where an illegal access occurred.<br>These bits retain the channel number (0 to 7) of the first illegal access after the DM0CMV.VF bit is cleared to 0.<br>If illegal access occurs while the DM0CMV.VF bit is 1, these bits do not change.<br>These bits are read-only and cannot be cleared. |
| 3 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | VF | Illegal access flag<br>This bit shows whether illegal access occurred in the DMAC0.<br>0: No illegal access has occurred in the DMAC0<br>1: Illegal access has occurred in the DMAC0<br>If illegal access occurs in the DMAC0 while this bit is 0, this bit is set, and DM0CMV.PEID[2:0], DM0CMV.SPID[1:0], DM0CMV.UM and DM0CMV.VCH[2:0] store their respective information.<br>If illegal access occurs in the DMAC0 while this bit is 1, this bit remains 1, and the contents of DM0CMV.PEID[2:0], DM0CMV.SPID[1:0], DM0CMV.UM and DM0CMV.VCH[2:0] do not change.<br>This bit can be cleared by using the CMVC register. |

### 8.8.2.4 DM1CMV — DMAC1 Register Access Protection Violation Register

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <DMA_base> + $0034_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | PEID2 | PEID1 | PEID0 | SPID1 | SPID0 | UM | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | VCH2 | VCH1 | VCH0 | — | — | — | VF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.18 DM1CMV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | When read, the value after reset is returned. |
| 22 to 17 | PEID[2:0]<br>SPID[1:0]<br>UM | Illegal access master information<br>These bits retain the accessing master information of the first illegal access after the DM1CMV.VF bit is cleared to 0.<br>If illegal access occurs while the DM1CMV.VF bit is 1, these bits do not change.<br>These bits are read-only and cannot be cleared. |
| 16 to 7 | Reserved | When read, the value after reset is returned. |
| 6 to 4 | VCH[2:0] | Channel where an illegal access occurred.<br>These bits retain the channel number (0 to 7) of the first illegal access after the DM1CMV.VF bit is cleared to 0.<br>If illegal access occurs while the DM1CMV.VF bit is 1, these bits do not change.<br>These bits are read-only and cannot be cleared. |
| 3 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | VF | Illegal access flag<br>This bit shows whether illegal access occurred in the DMAC1.<br>0: No illegal access has occurred in the DMAC1<br>1: Illegal access has occurred in the DMAC1<br>If illegal access occurs in the DMAC1 while this bit is 0, this bit is set, and DM1CMV.PEID[2:0], DM1CMV.SPID[1:0], DM1CMV.UM and DM1CMV.VCH[2:0] store their respective information.<br>If illegal access occurs in the DMAC1 while this bit is 1, this bit remains 1, and the contents of DM1CMV.PEID[2:0], DM1CMV.SPID[1:0], DM1CMV.UM and DM1CMV.VCH[2:0] do not change.<br>This bit can be cleared by using the CMVC register. |

### 8.8.2.5 CMVC — Register Access Protection Violation Clear Register

**Access:** This register can be read or written in 32-bit units.

**Address:** <DMA_base> + $003C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DM1VC | DM0VC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 8.19 CMVC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | DM1VC | DMAC1 illegal access flag clear<br>The DMAC1 illegal access flag (DM1CMV.VF) can be cleared by writing 1 to this bit. When read, this bit is always read as 0. |
| 0 | DM0VC | DMAC0 illegal access flag clear<br>The DMAC0 illegal access flag (DM0CMV.VF) can be cleared by writing 1 to this bit. When read, this bit is always read as 0. |

### 8.8.2.6 DMniCM — DMAC Channel Master Setting (ni = 00 to 07 and 10 to 17)

**Access:** This register can be read or written in 32-bit units.

**Address:** DM0iCM: <DMA_base> + $0100_H$ + $4_H$ × Ch. No. i (i = 0 to 7)
DM1iCM: <DMA_base> + $0120_H$ + $4_H$ × Ch. No. i (i = 0 to 7)

**Value after reset:** $0000\ 0010_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | PEID2 | PEID1 | PEID0 | SPID1 | SPID0 | UM | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R |

**Table 8.20 DMniCM Register Contents**

| Bit Position | Bit Name | Function ion |
|---|---|---|
| 31 to 7 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 6 to 4 | PEID[2:0] | Channel master PEID setting<br>Specifies the PEID information of the master assigned to the channel. |
| 3, 2 | SPID[1:0] | Channel master SPID setting<br>Specifies the SPID information used by the master assigned to the channel. |
| 1 | UM | Channel master UM setting<br>0: Enables access in supervisor mode.<br>1: Enables access in user mode and supervisor mode. |
| 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

**CAUTION**

DM00CM to DM07CM configure the channel master information of the DMAC0 channel 0 to 7 respectively.

DM10CM to DM17CM configure the channel master information of the DMAC1 channel 0 to 7 respectively.

For information about the functions this register offers, see **Section 8.6, Reliability Functions**.

# 8.9 DMAC Channel Registers

## 8.9.1 DMAC Channel Register Addresses

The DMAC channel registers are listed in the table below.

For <DMA_base>, see **Section 8.1.2, Register Base Address**.

**Table 8.21 DMAC Channel Register Addresses**

| Unit Name | Register Name | Register Symbol | Address | Access Permission | |
|---|---|---|---|---|---|
| | | | | Special Master | General Master |
| DMAC | DMAC source address | DSAm | <DMA_base> + $0400_H$ + $40_H$ × [channel number] | √ | √ |
| | DMAC destination address | DDAm | <DMA_base> + $0404_H$ + $40_H$ × [channel number] | √ | √ |
| | DMAC transfer count | DTCm | <DMA_base> + $0408_H$ + $40_H$ × [channel number] | √ | √ |
| | DMAC transfer control | DTCTm | <DMA_base> + $040C_H$ + $40_H$ × [channel number] | √ | √ |
| | DMAC reload source address | DRSAm | <DMA_base> + $0410_H$ + $40_H$ × [channel number] | √ | √ |
| | DMAC reload destination address | DRDAm | <DMA_base> + $0414_H$ + $40_H$ × [channel number] | √ | √ |
| | DMAC reload transfer count | DRTCm | <DMA_base> + $0418_H$ + $40_H$ × [channel number] | √ | √ |
| | DMAC channel operation enable setting | DCENm | <DMA_base> + $0420_H$ + $40_H$ × [channel number] | √ | √ |
| | DMAC transfer status | DCSTm | <DMA_base> + $0424_H$ + $40_H$ × [channel number] | √ | √ |
| | DMAC transfer status set | DCSTSm | <DMA_base> + $0428_H$ + $40_H$ × [channel number] | √ | √ |
| | DMAC transfer status clear | DCSTCm | <DMA_base> + $042C_H$ + $40_H$ × [channel number] | √ | √ |
| | DTFR setting | DTFRm | <DMA_base> + $0430_H$ + $40_H$ × [channel number] | √ | √ |
| | DTFR transfer request status | DTFRRQm | <DMA_base> + $0434_H$ + $40_H$ × [channel number] | √ | √ |
| | DTFR transfer request clear | DTFRRQCm | <DMA_base> + $0438_H$ + $40_H$ × [channel number] | √ | √ |

Note 1. The [channel number] in the offset addresses and “m” in the register symbols are numbers in the range from 0 to 15, and the correspondence is as follows.

| Channel number m | Channel |
|---|---|
| 0 | DMAC0 channel 0 |
| 1 | DMAC0 channel 1 |
| 2 | DMAC0 channel 2 |
| 3 | DMAC0 channel 3 |
| 4 | DMAC0 channel 4 |
| 5 | DMAC0 channel 5 |
| 6 | DMAC0 channel 6 |
| 7 | DMAC0 channel 7 |
| 8 | DMAC1 channel 0 |
| 9 | DMAC1 channel 1 |
| 10 | DMAC1 channel 2 |
| 11 | DMAC1 channel 3 |
| 12 | DMAC1 channel 4 |
| 13 | DMAC1 channel 5 |
| 14 | DMAC1 channel 6 |
| 15 | DMAC1 channel 7 |

## 8.9.2 Details of DMAC Channel Registers

The “m” in the register symbols indicates the DMA channel number (m = 0 to 15).

### 8.9.2.1 DSAm — DMAC Source Address Register

**Access:** This register can be read or written in 32-bit units.

**Address:** <DMA_base> + $0400_H$ + $40_H$ × Ch. No. m (m = 0 to 15)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SA31 | SA30 | SA29 | SA28 | SA27 | SA26 | SA25 | SA24 | SA23 | SA22 | SA21 | SA20 | SA19 | SA18 | SA17 | SA16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SA15 | SA14 | SA13 | SA12 | SA11 | SA10 | SA9 | SA8 | SA7 | SA6 | SA5 | SA4 | SA3 | SA2 | SA1 | SA0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 8.22 DSAm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | SA[31:0] | Source address<br>Specifies the DMA transfer source address. These bits are updated whenever a DMA cycle is executed. If you read from these bits, the transfer source address for the next DMA cycle is read. |

**CAUTIONS**

1. It is forbidden to write to these bits when the channel operation is enabled (DCENm.DTE bit = 1). If you do, the correct operation is not guaranteed.
2. If transfer error is generated in the read cycle of DMA transfer, the write cycle is not executed, but the source address is updated.
3. DMA transfer for misaligned data is not supported. The lower 4 bits of the address corresponding to each transfer data size is as follows. (× denotes an arbitrary bit.)
   The correct operation is not guaranteed if you set otherwise than the following table.

| Data Size | SA3 | SA2 | SA1 | SA0 |
|---|---|---|---|---|
| 8 bits | × | × | × | × |
| 16 bits | × | × | × | 0 |
| 32 bits | × | × | 0 | 0 |
| 64 bits | × | 0 | 0 | 0 |
| 128 bits | 0 | 0 | 0 | 0 |

### 8.9.2.2 DDAm — DMAC Destination Address Register

**Access:** This register can be read or written in 32-bit units.

**Address:** <DMA_base> + $0404_H$ + $40_H$ × Ch. No. m (m = 0 to 15)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DA31 | DA30 | DA29 | DA28 | DA27 | DA26 | DA25 | DA24 | DA23 | DA22 | DA21 | DA20 | DA19 | DA18 | DA17 | DA16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DA15 | DA14 | DA13 | DA12 | DA11 | DA10 | DA9 | DA8 | DA7 | DA6 | DA5 | DA4 | DA3 | DA2 | DA1 | DA0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 8.23 DDAm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DA[31:0] | Destination address<br>Specifies the DMA transfer destination address. These bits are updated whenever a DMA cycle is executed. If you read from these bits, the transfer destination address for the next DMA cycle is read. |

**CAUTIONS**

1. It is forbidden to write to these bits when the channel operation is enabled (DCENm.DTE bit = 1). If you do, the correct operation is not guaranteed.
2. If transfer error is generated in the read cycle of DMA transfer, the write cycle is not executed, but the destination address is updated.
3. DMA transfer for misaligned data is not supported. The lower 4 bits of the address corresponding to each transfer data size is as follows. (× denotes an arbitrary bit.)
   The correct operation is not guaranteed if you set otherwise than the following table.

| Data Size | DA3 | DA2 | DA1 | DA0 |
|---|---|---|---|---|
| 8 bits | × | × | × | × |
| 16 bits | × | × | × | 0 |
| 32 bits | × | × | 0 | 0 |
| 64 bits | × | 0 | 0 | 0 |
| 128 bits | 0 | 0 | 0 | 0 |

### 8.9.2.3 DTCm — DMAC Transfer Count Register

**Access:** This register can be read or written in 32-bit units.

**Address:** <DMA_base> + $0408_H$ + $40_H$ × Ch. No. m (m = 0 to 15)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ARC15 | ARC14 | ARC13 | ARC12 | ARC11 | ARC10 | ARC9 | ARC8 | ARC7 | ARC6 | ARC5 | ARC4 | ARC3 | ARC2 | ARC1 | ARC0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRC15 | TRC14 | TRC13 | TRC12 | TRC11 | TRC10 | TRC9 | TRC8 | TRC7 | TRC6 | TRC5 | TRC4 | TRC3 | TRC2 | TRC1 | TRC0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 8.24 DTCm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | ARC[15:0] | Address reload count<br>Specifies the number of transfers until the address reload when the reload function 2 is used, and also specifies the number of transfers when the block transfer 2 is used. If you read from those bits during DMA transfer, the address reload count for the next DMA cycle is read.<br>When the reload function 2 or block transfer 2 is used, DTCm.ARC[15:0] is decremented by one for every DMA cycle. When the reload function 2 or block transfer 2 is not used, DTCm.ARC[15:0] is not updated.<br>If the value is 0000H, it means that the number of transfers until the address reload when the reload function 2 is used and the number of transfers when the block transfer 2 is used are 65536. |
| 15 to 0 | TRC[15:0] | Transfer count<br>Configures the number of transfers. DTCm.TRC[15:0] is decremented by one whenever a DMA cycle is executed. If you read from those bits, the remaining number of transfers for the next DMA cycle is read. If the reload function is not used, after the last transfer is complete, the value at the completion (0000H) is retained. |

| TRC15-0 | Operation |
|---|---|
| $0000_H$ | The number of transfers is 65536, or the transfer is complete. |
| $0001_H$ | The number of transfers is 1, or remaining transfer count is 1. |
| : | : |
| $FFFF_H$ | The number of transfers is 65535, or remaining transfer count is 65535. |

**CAUTIONS**

1. It is forbidden to write to those bits when the channel operation is enabled (DCENm.DTE bit = 1). If you do, the correct operation is not guaranteed.
2. If transfer error is generated in the read cycle of DMA transfer, the write cycle is not executed, but the transfer count and the address reload count are updated.

### 8.9.2.4 DTCTm — DMAC Transfer Control Register

**Access:** This register can be read or written in 32-bit units.

**Address:** <DMA_base> + $040C_H$ + $40_H$ × Ch. No. m (m = 0 to 15)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | ESE | DRS | — | — | — | — | — | CHNSEL2 | CHNSEL1 | CHNSEL0 | CHNE1 | CHNE0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | TCE | MLE | RLD2M1 | RLD2M0 | RLD1M1 | RLD1M0 | DACM1 | DACM0 | SACM1 | SACM0 | DS2 | DS1 | DS0 | TRM1 | TRM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 8.25 DTCTm Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 27 | ESE | DMA transfer disable on transfer error setting<br>Specifies whether to execute a DMA cycle when the DCSTm.ER bit is set due to DMA transfer error.<br>If this bit is cleared to 0, even when the DCSTm.ER bit is set due to DMA transfer error, the following DMA cycles can be executed. If this bit is set to 1, the following DMA cycles are not executed when the DCSTm.ER bit is set due to DMA transfer error.<br>0: DMA cycles are executed while the DCSTm.ER bit is set.<br>1: DMA cycles are not executed while the DCSTm.ER bit is set. |
| 26 | DRS | DMA transfer request selection assignment<br>Selects the type of DMA transfer requests to be accepted.<br>0: Software DMA transfer request<br>1: Hardware DMA transfer request |
| 25 to 21 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 20 to 18 | CHNSEL[2:0] | Selection of next channel in the chain<br>Specifies the next channel in the chain.<br>The next channel must be another channel in the same DMAC. You cannot specify a channel in the different DMAC. It is prohibited to specify the channel as the next channel in the chain. (If you do, the correct operation is not guaranteed.) |
| 17, 16 | CHNE[1:0] | Chain enable<br>Selects the chain function.<br>00: Disabled<br>01: Chain at the last transfer<br>A chain request is generated at the completion of the DMA cycle when the remaining transfer count is one.<br>10: Setting prohibited. (The operation is not guaranteed.)<br>11: Always chain<br>A chain request is generated at the completion of every DMA cycle. |
| 15 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 14 | TCE | Transfer completion interrupt enable<br>If this bit is set, a transfer completion interrupt is generated at the completion of the last transfer. |

**Table 8.25 DTCTm Register Contents (2/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 13 | MLE | Continuous transfer enable<br>If this bit is set, the DCENm.DTE bit is not cleared at the completion of DMA transfer. Even if the DCSTm.TC bit is not cleared, DMA transfer starts when there is a DMA transfer request.<br>0: The DCENm.DTE bit is cleared at the completion of DMA transfer. The next DMA transfer starts only after the DCSTm.TC bit is cleared.<br>1: The DCENm.DTE bit is not cleared at the completion of DMA transfer. Even if the DCSTm.TC bit is not cleared, DMA transfer starts when there is a DMA transfer request. |
| 12, 11 | RLD2M[1:0] | Reload function 2 setting<br>Specifies the reload function 2.<br>00: Reload function 2 is disabled.<br>01: Reload function 2 is enabled.<br>The source address and address reload count are reloaded at the completion of the DMA cycle in which the address reload count is 1.<br>10: Reload function 2 is enabled.<br>The destination address and address reload count are reloaded at the completion of the DMA cycle in which the address reload count is 1.<br>11: Reload function 2 is enabled.<br>The source address, destination address, and address reload count are reloaded at the completion of the DMA cycle in which the address reload count is 1. |
| 10, 9 | RLD1M[1:0] | Reload function 1 setting<br>Specifies the reload function 1.<br>00: Reload function 1 is disabled.<br>01: Reload function 1 is enabled.<br>The source address and transfer count are reloaded at the completion of the DMA cycle in which the remaining transfer count is 1. (If the reload function 2 is enabled, the address reload count is also reloaded.)<br>10: Reload function 1 is enabled.<br>The destination address and transfer count are reloaded at the completion of the DMA cycle in which the remaining transfer count is 1. (If the reload function 2 is enabled, the address reload count is also reloaded.)<br>11: Reload function 1 is enabled.<br>The source address, destination address, and transfer count are reloaded at the completion of the DMA cycle in which the remaining transfer count is 1. (If the reload function 2 is enabled, the address reload count is also reloaded.) |
| 8, 7 | DACM[1:0] | Destination address count direction<br>Specifies the count direction of the destination address.<br>**DACM1 / DACM0 / Direction of Count**<br>0 / 0 / Increment<br>0 / 1 / Decrement<br>1 / 0 / Fixed<br>1 / 1 / Setting prohibited (The operation is not guaranteed.) |
| 6, 5 | SACM[1:0] | Source address count direction<br>Specifies the count direction of the source address.<br>**SACM1 / SACM0 / Direction of Count**<br>0 / 0 / Increment<br>0 / 1 / Decrement<br>1 / 0 / Fixed<br>1 / 1 / Setting prohibited (The operation is not guaranteed.) |

**Table 8.25 DTCTm Register Contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 4 to 2 | DS[2:0] | Transfer data size<br>Specifies the transfer data size. |
| 1, 0 | TRM[1:0] | Transfer mode<br>Specifies the DMA transfer mode.<br>00: Single transfer<br>01: Block transfer 1 (The number of transfers is specified by the transfer count.)<br>10: Block transfer 2 (The number of transfers is specified by the address reload count.)<br>11: Setting prohibited (The operation is not guaranteed.) |

| DS2 | DS1 | DS0 | Transfer Data Size |
|---|---|---|---|
| 0 | 0 | 0 | 8 bits |
| 0 | 0 | 1 | 16 bits |
| 0 | 1 | 0 | 32 bits |
| 0 | 1 | 1 | 64 bits |
| 1 | 0 | 0 | 128 bits |
| Other than the above | | | Setting prohibited (The operation is not guaranteed.) |

**CAUTIONS**

1. Except for the case to clear DTCTm.MLE bit, it is forbidden to write to those bits when the channel operation is enabled (DCENm.DTE bit = 1). If you do, the correct operation is not guaranteed.
2. If forbidden settings are used for each bits, the correct operation is not guaranteed.

### 8.9.2.5 DRSAm — DMAC Reload Source Address Register

**Access:** This register can be read or written in 32-bit units.

**Address:** <DMA_base> + $0410_H$ + $40_H$ × Ch. No. m (m = 0 to 15)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSA31 | RSA30 | RSA29 | RSA28 | RSA27 | RSA26 | RSA25 | RSA24 | RSA23 | RSA22 | RSA21 | RSA20 | RSA19 | RSA18 | RSA17 | RSA16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSA15 | RSA14 | RSA13 | RSA12 | RSA11 | RSA10 | RSA9 | RSA8 | RSA7 | RSA6 | RSA5 | RSA4 | RSA3 | RSA2 | RSA1 | RSA0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 8.26 DRSAm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | RSA[31:0] | Reload source address<br>Specifies the source address to be reloaded to the DMA source address register when the reload function 1 or reload function 2 is used. |

**CAUTION**

DMA transfer for misaligned data is not supported. The lower 4 bits of the address corresponding to each transfer data size is as follows. (× denotes an arbitrary bit.) The correct operation is not guaranteed if you set otherwise than the following table.

| Data Size | RSA3 | RSA2 | RSA1 | RSA0 |
|---|---|---|---|---|
| 8 bits | × | × | × | × |
| 16 bits | × | × | × | 0 |
| 32 bits | × | × | 0 | 0 |
| 64 bits | × | 0 | 0 | 0 |
| 128 bits | 0 | 0 | 0 | 0 |

### 8.9.2.6 DRDAm — DMAC Reload Destination Address Register

**Access:** This register can be read or written in 32-bit units.

**Address:** <DMA_base> + $0414_H$ + $40_H$ × Ch. No. m (m = 0 to 15)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDA31 | RDA30 | RDA29 | RDA28 | RDA27 | RDA26 | RDA25 | RDA24 | RDA23 | RDA22 | RDA21 | RDA20 | RDA19 | RDA18 | RDA17 | RDA16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDA15 | RDA14 | RDA13 | RDA12 | RDA11 | RDA10 | RDA9 | RDA8 | RDA7 | RDA6 | RDA5 | RDA4 | RDA3 | RDA2 | RDA1 | RDA0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 8.27 DRDAm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | RDA[31:0] | Reload destination address<br>Specifies the destination address to be reloaded to the DMA destination address register when the reload function 1 or reload function 2 is used. |

**CAUTION**

DMA transfer for misaligned data is not supported. The lower 4 bits of the address corresponding to each transfer data size is as follows. (× denotes an arbitrary bit.) The correct operation is not guaranteed if you set otherwise than the following table.

| Data Size | RDA3 | RDA2 | RDA1 | RDA0 |
|---|---|---|---|---|
| 8 bits | × | × | × | × |
| 16 bits | × | × | × | 0 |
| 32 bits | × | × | 0 | 0 |
| 64 bits | × | 0 | 0 | 0 |
| 128 bits | 0 | 0 | 0 | 0 |

### 8.9.2.7 DRTCm — DMAC Reload Transfer Count Register

**Access:** This register can be read or written in 32-bit units.

**Address:** <DMA_base> + $0418_H$ + $40_H$ × Ch. No. m (m = 0 to 15)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RARC15 | RARC14 | RARC13 | RARC12 | RARC11 | RARC10 | RARC9 | RARC8 | RARC7 | RARC6 | RARC5 | RARC4 | RARC3 | RARC2 | RARC1 | RARC0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RTRC15 | RTRC14 | RTRC13 | RTRC12 | RTRC11 | RTRC10 | RTRC9 | RTRC8 | RTRC7 | RTRC6 | RTRC5 | RTRC4 | RTRC3 | RTRC2 | RTRC1 | RTRC0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 8.28 DRTCm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | RARC[15:0] | Reload address reload count<br>Specifies the value to be reloaded to the address reload count in the transfer count register at the timing of reload when the reload function 2 is used. |
| 15 to 0 | RTRC[15:0] | Reload transfer count<br>Specifies the value to be reloaded to the transfer count in the transfer count register at the timing of reload when the reload function 1 is used. |

### 8.9.2.8 DCENm — DMAC Channel Operation Enable Setting Register

**Access:** This register can be read or written in 32-bit units.

**Address:** <DMA_base> + $0420_H$ + $40_H$ × Ch. No. m (m = 0 to 15)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DTE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 8.29 DCENm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | DTE | Channel operation enable<br>Specifies whether to enable or disable the transfer operation of the channel. If the DCENm.DTE bit is 1, DMA transfer starts when there is a DMA transfer request. If the DTCTm.MLE bit is 0, this bit is automatically cleared when DMA transfer is completed. In addition, if 0 is written to the DCENm.DTE bit during DMA transfer, the DMA transfer is suspended. If 1 is written to the DCENm.DTE bit during suspension, the suspension is cleared and the DMA transfer resumes.<br>0: Channel operation is disabled/Channel suspended<br>1: Channel operation is enabled/Channel suspension cleared |

### 8.9.2.9 DCSTm — DMAC Transfer Status Register

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <DMA_base> + $0424_H$ + $40_H$ × Ch. No. m (m = 0 to 15)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | ERWR | — | — | CY | ER | — | — | TC | — | — | DR | SR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.30 DCSTm Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 12 | Reserved | When read, the value after reset is returned. |
| 11 | ERWR | DMA Transfer Error occurring cycle<br>This bit is updated at the same time as setting of the DMA transfer error flag (DCSTm.ER), indicating in which cycle of read or write the DMA transfer error occurred. This bit is not updated when a new DMA transfer error occurs after the DCSTm.ER bit has been set. If the DCSTm.ER bit is cleared, this bit is also cleared to 0.<br>0: DMA transfer error occurs in the read cycle.<br>1: DMA transfer error occurs in the write cycle. |
| 10, 9 | Reserved | When read, the value after reset is returned. |
| 8 | CY | DMA cycle execution state<br>This bit shows whether a DMA cycle is ongoing in this channel.<br>0: DMA cycle is not ongoing.<br>1: DMA cycle is ongoing. |
| 7 | ER | Transfer error flag<br>This bit is set when DMA transfer error is generated. If this bit is 1 and the DTCTm.ESE bit is set, a DMA cycle is not executed even when a DMA transfer request is generated.<br>0: No DMA transfer error is generated<br>1: DMA transfer error is generated |
| 6, 5 | Reserved | When read, the value after reset is returned. |
| 4 | TC | Transfer completion flag<br>This bit is set at the completion of the last transfer and indicates that the DMA transfer is complete. If the DTCTm.MLE bit is 0 and this bit is 1, a DMA cycle is not executed when a DMA transfer request is generated.<br>0: DMA transfer is incomplete<br>1: DMA transfer is complete |
| 3, 2 | Reserved | When read, the value after reset is returned. |
| 1 | DR | Hardware DMA transfer request status<br>This bit shows whether there is a hardware DMA transfer request (DMARQ) from the DTFR.<br>This bit changes regardless of the value of the DCENm.DTE bit when a hardware DMA transfer request from the DTFR is generated. If the software DMA transfer request has been selected in the transfer request selection bit (DTCTm.DRS) in the DMAC transfer control register, this bit is not set even when a hardware DMA transfer request is input from the DTFR.<br>0: There is no hardware DMA transfer request<br>1: There is a hardware DMA transfer request |

Table 8.30 DCSTm Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | SR | Software DMA transfer request flag<br>This bit shows whether there is a software DMA transfer request. This bit is automatically cleared when executing the DMA transfer. A user can set this bit by writing 1 to the DCSTSm.SRS bit in the DMAC transfer status set register (DCSTSm). In addition, a user can clear this bit by writing 1 to the DCSTCm.SRC bit in the DMAC transfer status clear register (DCSTCm), but if this is done, the ongoing DMA transfer is aborted and cannot be resumed.<br>0: There is no software DMA transfer request<br>1: There is a software DMA transfer request |

### 8.9.2.10 DCSTSm — DMAC Transfer Status Set Register

**Access:** This register can be read or written in 32-bit units.

**Address:** <DMA_base> + $0428_H$ + $40_H$ × Ch. No. m (m = 0 to 15)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SRS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 8.31 DCSTSm Set Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | SRS | Software DMA transfer request flag<br>A user can set the software DMA transfer request flag (DCSTm.SR) by writing 1 to this bit. When read, this bit is always read as 0. |

### 8.9.2.11 DCSTCm — DMAC Transfer Status Clear Register

**Access:** This register can be read or written in 32-bit units.

**Address:** <DMA_base> + $042C_H$ + $40_H$ × Ch. No. m (m = 0 to 15)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ERC | — | — | TCC | — | — | — | SRC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R | R | R/W | R | R | R | R/W |

**Table 8.32 DCSTCm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 7 | ERC | Transfer error flag clear<br>The DMA transfer error flag (DCSTm.ER) can be cleared by writing 1 to this bit. When read, this bit is always read as 0.. |
| 6, 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 | TCC | Transfer completion flag clear<br>The transfer completion flag (DCSTm.TC) can be cleared by writing 1 to this bit. When read, this bit is always read as 0.. |
| 3 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | SRC | Software DMA transfer request flag clear<br>The software DMA transfer request flag (DCSTm.SR) can be cleared by writing 1 to this bit. When read, this bit is always read as 0.. |

### 8.9.2.12 DTFRm — DTFR Setting Register

**Access:** This register can be read or written in 32-bit units.

**Address:** <DMA_base> + $0430_H$ + $40_H$ × Ch. No. m (m = 0 to 15)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | REQSEL6 | REQSEL5 | REQSEL4 | REQSEL3 | REQSEL2 | REQSEL1 | REQSEL0 | REQEN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 8.33 DTFRm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 7 to 1 | REQSEL[6:0] | Hardware DMA transfer source selection<br>Selects one out of 128 hardware DMA transfer sources as the hardware DMA transfer request.<br>000_0000: Select the DMACTRG[0] input<br>:<br>111_1111: Select the DMACTRG[127] input |
| 0 | REQEN | Hardware DMA transfer source selection enable<br>This bit enables/disables the hardware DMA transfer source selection.<br>0: Hardware DMA transfer source selection is disabled.<br>1: Hardware DMA transfer source selection is enabled.<br>If this bit is 0, even when the hardware DMA transfer source selected by the DTFRm.REQSEL[6:0] bits is activated, it is not recognized as a hardware DMA transfer request, and a hardware DMA transfer request is not generated. |

### 8.9.2.13 DTFRRQm — DTFR Transfer Request Status Register

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <DMA_base> + $0434_H$ + $40_H$ × Ch. No. m (m = 0 to 15)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.34 DTFRRQm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | DRQ | Hardware DMA transfer request status<br>This bit indicates that a hardware DMA transfer request exists or is retained.<br>• This bit indicates whether a hardware DMA transfer request is retained or not. When the DMA transfer request acceptance signal from the DMAC is asserted, this bit is automatically cleared. A user can clear this bit by writing 1 to the DTFRRQCm.DRQC bit.<br>This bit changes regardless of the value of the DTFRm.REQEN bit when a hardware DMA transfer request from the outside is generated.<br>0: There is no hardware DMA transfer request<br>1: There is a hardware DMA transfer request |

### 8.9.2.14 DTFRRQCm — DTFR Transfer Request Clear Register

**Access:** This register can be read or written in 32-bit units.

**Address:** <DMA_base> + $0438_H$ + $40_H$ × Ch. No. m (m = 0 to 15)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DRQC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 8.35 DTFRRQCm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | DRQC | Hardware DMA transfer request clear<br>A user can clear the DTFRRQm.DRQ bit by writing 1 to this bit.<br>When read, this bit is always read as 0. |

# Section 9 Reset Controller

## 9.1 Overview

Several system reset functions are provided in order to initialize CPU core and peripheral functions as well as their associated registers.

A reset can be caused by the following events:

- External reset ($\overline{\text{RESET}}$)
- Power-on-clear (POCRES)
- Watchdog timer reset (WDTA0RES, WDTA1RES)
- Clock monitor reset ($\overline{\text{CLMA0RES}}$, $\overline{\text{CLMA1RES}}$, $\overline{\text{CLMA2RES}}$)
- Low-voltage indicator reset ($\overline{\text{LVIRES}}$)
- Software reset (SWRES)
- Debugger reset ($\overline{\text{DBRES}}$)
- Core voltage monitor reset ($\overline{\text{CVMRES}}$)
- Transition to DeepSTOP mode

### 9.1.1 Reset Sources

Reset levels and reset sources are shown below.

Various reset sources are assigned to the different levels of the reset.

**Table 9.1 Reset Sources and Reset Targets**

| Reset Level | Reset Source | Clock Generator (except PLL)/ Real-Time Clock/CVM/LVI | Always-On Area Modules*1 | Isolated area Modules*2 |
|---|---|---|---|---|
| 1 | Power-on-clear (POCRES)<br>Debugger reset ($\overline{\text{DBRES}}$) | Reset | Reset | Reset |
| 2 | External reset ($\overline{\text{RESET}}$)<br>Watchdog timer reset (WDTA0RES, WDTA1RES)<br>Clock monitor reset ($\overline{\text{CLMA0RES}}$, $\overline{\text{CLMA1RES}}$, $\overline{\text{CLMA2RES}}$)<br>Core voltage monitor reset ($\overline{\text{CVMRES}}$)<br>Low voltage indicator reset ($\overline{\text{LVIRES}}$)<br>Software reset (SWRES) | Not reset target*3 | Reset | Reset |
| 3 | Reset by DeepSTOP mode | Not reset target | Not reset target | Reset |

Note 1. Clock generator, real-time clock, CVM, and LVI are excluded.
Note 2. PLL is included.
Note 3. In clock monitor reset, oscillator-related registers for clock monitoring are initialized.

Reset level 1: Initializes the entire microcontroller.

Reset level 2: For a quick return to normal operating mode by eliminating the oscillator stabilization time, initializes the entire microcontroller except for the clock generator and the real-time clock.

Reset level 3: At the transition to DeepSTOP mode, initializes all the Isolated area (ISO area).

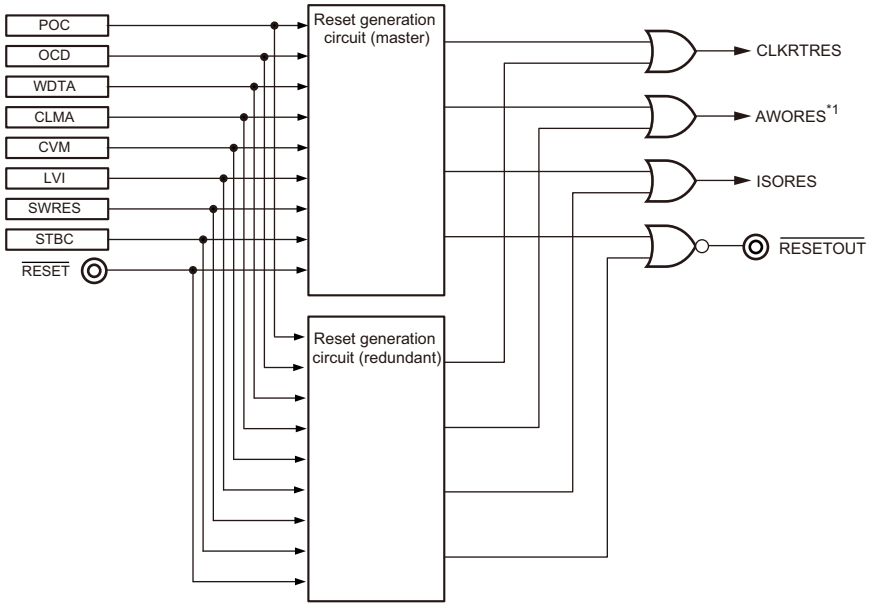
If Power-on-clear (POCRES), Core voltage monitor reset (CVMRES), or Clock monitor reset (CLMA0RES, CLMA1RES, CLMA2RES) generates while writing to Retention RAM, Retention RAM data is not retained.

If reset source other than shown above is generated while writing to Retention RAM, Retention RAM data can be retained. In this case, data value will be before or after writing.

### 9.1.2 Reset Controller Redundancy

The reset controller of the microcontroller has a redundant configuration, and includes duplicated reset generation circuits. Such configuration enables initialization of the reset targeted area without failure even if one of the two reset generation circuits fails.

The configuration of the reset generation circuits is shown in the figure below.




Note 1. Real-time clock is excluded.

**Figure 9.1 Reset Controller Redundancy**

At the generation of a reset, the same reset source signal is input to two reset generation circuits.

According to the reset source, the two reset generation circuits output the Always-On area reset signal (AWORES), Isolated area reset signal (ISORES), clock generator/real-time clock reset signal (CLKRTRES), and $\overline{\text{RESETOUT}}$ signal.

The AWORES, ISORES, CLKRTRES, and $\overline{\text{RESETOUT}}$ signals are generated by executing the logical OR of the signals output from two reset generation circuits. Thus, a reset signal is generated normally even if one of the two reset generation circuits fails.

Whether a reset generation circuit operates normally can be checked by reading and comparing the reset factor registers of the respective reset generation circuits.

### 9.1.3 Reset Output ($\overline{\text{RESETOUT}}$)

When a reset source of reset level 1 or 2 is generated, a reset output signal ($\overline{\text{RESETOUT}}$) is output to the outside. Reset output is used to reset external devices at the same time as a reset is generated inside the microcontroller.

For details, see **Section 2.11.1.1, P8_6: $\overline{\text{RESETOUT}}$**.

### 9.1.4 Reset Flag

To identify a reset source, two registers with a flag for each reset source are provided. The main elements of the reset controller are shown in **Figure 9.2, Block Diagram of the Reset Controller**.

### 9.1.5 Clock Supply

The clock supply to the reset controller is shown in the following table.

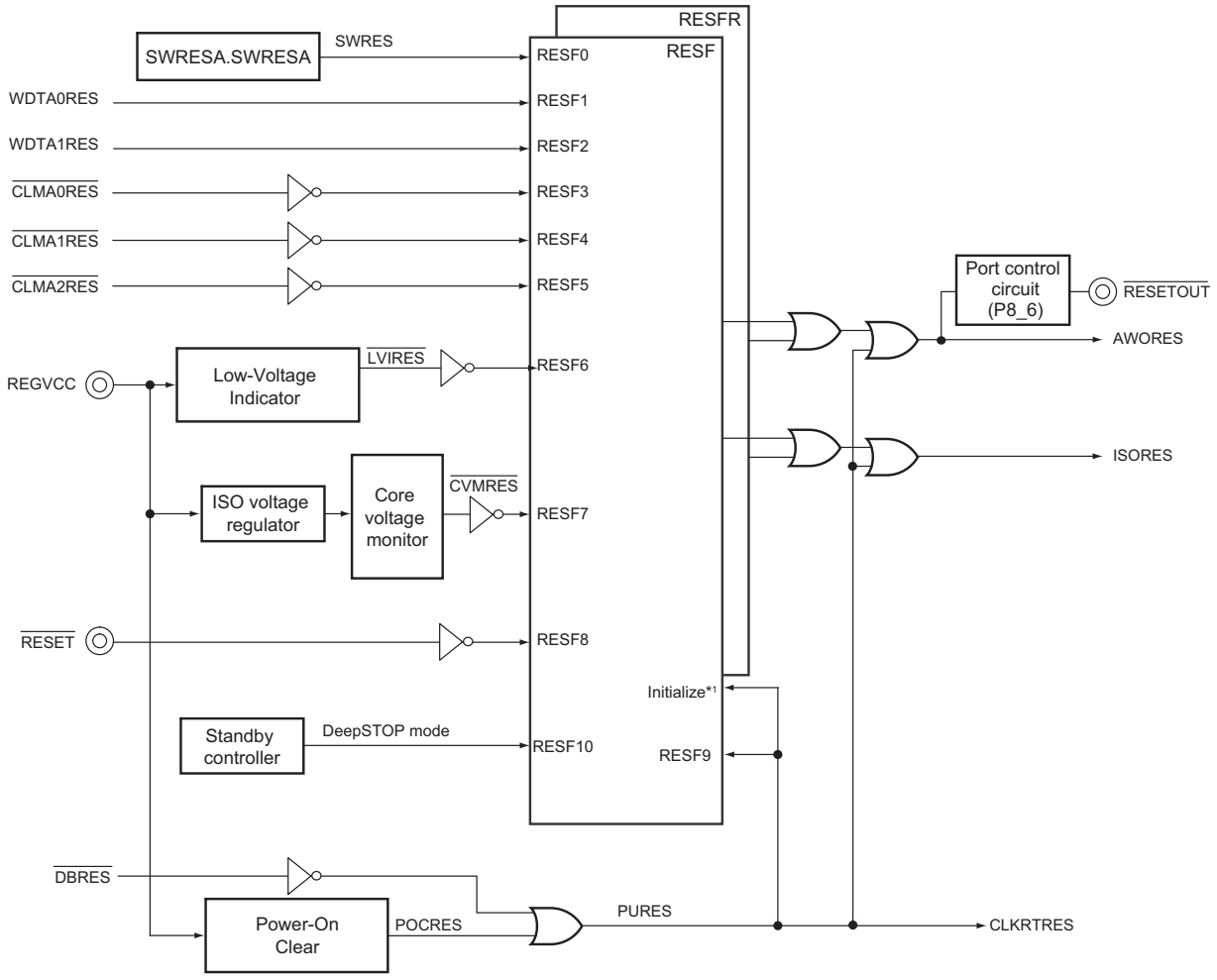
**Table 9.2 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name |
|---|---|---|
| Reset | Register access clock | CPUCLK4 |

## 9.2 Configuration

### 9.2.1 Block Diagram

A block diagram of the reset controller is shown below.




Note 1. Registers RESF and RESFR are initialized by PURES.
After an initialization, bits RESF9 and RESFR9 are set to 1.

**Figure 9.2 Block Diagram of the Reset Controller**

### (1) Reset signals

The reset controller manages the generation of three reset signals upon occurrence of reset signals from various reset sources:

- Always-On area (AWO area) reset (AWORES)
  AWORES is generated by all reset sources except the transition to DeepSTOP mode.
  AWORES resets all modules in the Always-On area (AWO area) except clock generation circuit, real-time clock, core voltage monitor, and low-voltage detection circuit.
- Isolated area (ISO area) reset (ISORES)
  ISORES is generated by all reset sources.
  ISORES resets all modules (including PLL) in the Isolated area (ISO area).
- CLKRTRES
  CLKRTRES is generated by the power-on clear or debugger reset sources.
  CLKRTRES resets the clock generation circuit (excluding PLL) and real-time clock.

The power-up reset (PURES) is generated by the power-on clear and debugger reset sources.

Following the generation of an AWORES reset, with the exception of the PLL, all clock-generation circuits that were operating at the time (LS IntOSC, HS IntOSC, MainOSC, SubOSC) continue to operate. On the generation of a $\overline{\text{CLMA0RES}}$ reset, the HS IntOSC that was the target for CLMA0 monitoring is reset. On the generation of a $\overline{\text{CLMA1RES}}$ reset, the MainOSC that was the target for CLMA1 monitoring is reset.

The PURES initializes all of the clock generation circuits. It is necessary to restart the clock generation circuit after recovery from the PURES.

The CPU reset is the Isolated area (ISO area) reset (ISORES) to the CPU sub system.

### (2) Reset flags

The reset factor register (RESF) and the redundant reset factor register (RESFR) hold a flag for each reset source, and this flag is set when the corresponding reset is asserted.

All reset flags except RESF9 and RESFR9 are initialized by a power-up reset (PURES). (Bits RESF9 and RESFR9 are set to 1 after initialization.) In addition, all the bits can be cleared by software.

For details, see **Section 9.1.4, Reset Flag**.

### (3) On-chip module resets

#### (a) Watchdog timer resets

The watchdog timers 0, 1 can generate two types of resets: WDTA0RES and WDTA1RES.
For details, see **Section 9.4.6, Watchdog Timer (WDTA) Reset**.

#### (b) Clock monitor resets

The clock monitors can generate three resets: $\overline{\text{CLMA0RES}}$, $\overline{\text{CLMA1RES}}$, and $\overline{\text{CLMA2RES}}$.
For details, see **Section 9.4.8, Clock Monitor (CLMA) Reset**.

#### (c) Debugger reset

A reset is generated by a command from a debugger. This leads to a generation of power-up reset PURES. For details, see **Section 9.4.9, Debugger Reset**.

### (4) Software controlled reset (SWRES)

A software reset SWRES can be generated by use of the software reset register SWRESA.

For details, see **Section 9.4.7, Software Reset**.

### (5) Reset output signal

During reset and after release from the reset, port P8_6 outputs low level as $\overline{\text{RESETOUT}}$ function. For details, see **Section 2.11.1.1, P8_6: $\overline{\text{RESETOUT}}$**.

### (6) Power supply monitoring

The following power supply detection circuits observe the level of the external power supply REGVCC.

#### (a) Low-Voltage Indicator

The low-voltage indicator (LVI) generates the $\overline{\text{LVIRES}}$ reset, if the voltage level of REGVCC drops below a certain level. The level can be adjusted and the $\overline{\text{LVIRES}}$ can be masked.
For details, see **Section 9.4.3, Low-Voltage Indicator (LVI) Reset**.

#### (b) Power-On-Clear

The power-on-clear circuit (POC) continuously compares the power supply voltage REGVCC with an internal reference voltage. Thus, a reset is generated when the power supply voltage goes below a certain level.
For details, see **Section 9.4.2, Power-On-Clear (POC) Reset**.

#### (c) Core Voltage Monitor

A reset can be generated when the core voltage monitor (CVM) detects over- or undervoltage in core voltage. (Output/not output can be set by option byte.)
For details, see **Section 9.4.4, Core Voltage Monitor (CVM) Reset**.

### (7) Masking of reset sources in debugging mode

The following reset sources can be masked during debugging:

**Table 9.3 Reset Sources to be Masked during Debugging**

| Reset Source | Maskable/Non-maskable |
|---|---|
| Power-on clear (POCRES) | — |
| Debugger reset ($\overline{\text{DBRES}}$) | — |
| External reset ($\overline{\text{RESET}}$) | √ |
| Low-voltage indicator reset ($\overline{\text{LVIRES}}$) | √ |
| Clock monitor reset ($\overline{\text{CLMA0RES}}$, $\overline{\text{CLMA1RES}}$, $\overline{\text{CLMA2RES}}$) | √ |
| Watchdog timer reset (WDTA0RES, WDTA1RES) | √ |
| Core voltage monitor reset ($\overline{\text{CVMRES}}$) | √ |
| Software reset (SWRES) | √ |
| Reset by DeepSTOP mode | — |

## 9.3 Registers

This section contains a description of all registers of the reset controller.

### 9.3.1 Reset Controller Registers Overview

The reset controller is controlled and operated by the following registers:

**Table 9.4 Reset Controller Registers Overview**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| **RESCTL** | **Reset flag registers** | | |
| | Reset factor register | RESF | FFF8 $0760_H$ |
| | Reset factor clear register | RESFC | FFF8 $0768_H$ |
| | Redundant reset factor register | RESFR | FFF8 $0860_H$ |
| | Redundant reset factor clear register | RESFCR | FFF8 $0868_H$ |
| | **Software reset control register** | | |
| | Software reset register | SWRESA | FFF8 $0A04_H$ |
| | **Cyclic RUN mode reset vector address register** | | |
| | Cyclic RUN mode RBASE register | CYCRBASE | FFF8 $3600_H$ |

**NOTES**

1. For the LVI related, RAM store related, and CVM related registers, see **Section 11, Supply Voltage Monitor**.
2. As for the protection registers, see **Section 5, Write-Protected Registers**.

## 9.3.2 Details of Reset Flag Registers

### 9.3.2.1 RESF — Reset Factor Register

This register contains information about which type of resets occurred after the last power-on clear reset. This register is initialized by a power-up reset PURES.

Each reset condition sets the corresponding flag in the register.

For example, if a clock monitor reset $\overline{\text{CLMA0RES}}$ occurs after a watchdog timer reset WDTA0RES, RESF reads 0000 000A$_H$.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 0760$_H$

**Value after reset:** 0000 0200$_H$ / 0000 0300$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | RESF10 | RESF9 | RESF8 | RESF7 | RESF6 | RESF5 | RESF4 | RESF3 | RESF2 | RESF1 | RESF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1/0*1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. For details, see **Figure 9.4, When $\overline{\text{RESET}}$ is Released before Execution of Flash Sequence**.

**Table 9.5 RESF Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When read, the value after reset is returned. |
| 10 | RESF10 | Reset flag by DeepSTOP mode<br>0: No reset occurred<br>1: Reset has occurred |
| 9 | RESF9 | Power-up reset flag<br>0: No reset occurred<br>1: Reset has occurred |
| 8 | RESF8 | External reset flag<br>0: No reset occurred<br>1: Reset has occurred |
| 7 | RESF7 | CVM reset flag<br>0: No reset occurred<br>1: Reset has occurred |
| 6 | RESF6 | LVI reset flag<br>0: No reset occurred<br>1: Reset has occurred |
| 5 | RESF5 | CLMA2 reset flag<br>0: No reset occurred<br>1: Reset has occurred |
| 4 | RESF4 | CLMA1 reset flag<br>0: No reset occurred<br>1: Reset has occurred |
| 3 | RESF3 | CLMA0 reset flag<br>0: No reset occurred<br>1: Reset has occurred |

Table 9.5 RESF Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 2 | RESF2 | WDTA1 reset flag<br>0: No reset occurred<br>1: Reset has occurred |
| 1 | RESF1 | WDTA0 reset flag<br>0: No reset occurred<br>1: Reset has occurred |
| 0 | RESF0 | Software reset flag<br>0: No reset occurred<br>1: Reset has occurred |

### 9.3.2.2 RESFC — Reset Factor Clear Register

This register clears the reset flags of the RESF register.

**Access:** This register is a write-only register that can be written in 32-bit units.

**Address:** FFF8 0768$_H$

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | RESFC10 | RESFC9 | RESFC8 | RESFC7 | RESFC6 | RESFC5 | RESFC4 | RESFC3 | RESFC2 | RESFC1 | RESFC0 |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | W | W | W | W | W | W | W | W | W | W | W |

**Table 9.6 RESFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When writing, write “0”. |
| 10 | RESFC10 | Reset flag clear by DeepSTOP mode<br>0: Do not clear flag<br>1: Clear flag |
| 9 | RESFC9 | Power-up reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 8 | RESFC8 | External reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 7 | RESFC7 | CVM reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 6 | RESFC6 | LVI reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 5 | RESFC5 | CLMA2 reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 4 | RESFC4 | CLMA1 reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 3 | RESFC3 | CLMA0 reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 2 | RESFC2 | WDTA1 reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 1 | RESFC1 | WDTA0 reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 0 | RESFC0 | Software reset flag clear<br>0: Do not clear flag<br>1: Clear flag |

### 9.3.2.3 RESFR — Redundant Reset Factor Register

This register is a duplication of the reset factor register. This register is initialized by a power-up reset PURES.

In accordance with the setting conditions for each bit in the reset factor register, the same bits are set in this register.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 $0860_H$

**Value after reset:** $0000\ 0200_H$ / $0000\ 0300_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | RESFR10 | RESFR9 | RESFR8 | RESFR7 | RESFR6 | RESFR5 | RESFR4 | RESFR3 | RESFR2 | RESFR1 | RESFR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1/0*1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. For details, see **Figure 9.4, When $\overline{\text{RESET}}$ is Released before Execution of Flash Sequence**.

**Table 9.7 RESFR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When read, the value after reset is returned. |
| 10 | RESFR10 | Reset flag by DeepSTOP mode<br>0: No reset occurred<br>1: Reset has occurred |
| 9 | RESFR9 | Power-up reset flag<br>0: No reset occurred<br>1: Reset has occurred |
| 8 | RESFR8 | External reset flag<br>0: No reset occurred<br>1: Reset has occurred |
| 7 | RESFR7 | CVM reset flag<br>0: No reset occurred<br>1: Reset has occurred |
| 6 | RESFR6 | LVI reset flag<br>0: No reset occurred<br>1: Reset has occurred |
| 5 | RESFR5 | CLMA2 reset flag<br>0: No reset occurred<br>1: Reset has occurred |
| 4 | RESFR4 | CLMA1 reset flag<br>0: No reset occurred<br>1: Reset has occurred |
| 3 | RESFR3 | CLMA0 reset flag<br>0: No reset occurred<br>1: Reset has occurred |
| 2 | RESFR2 | WDTA1 reset flag<br>0: No reset occurred<br>1: Reset has occurred |
| 1 | RESFR1 | WDTA0 reset flag<br>0: No reset occurred<br>1: Reset has occurred |

Table 9.7 RESFR Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | RESFR0 | Software reset flag<br>0: No reset occurred<br>1: Reset has occurred |

### 9.3.2.4 RESFCR — Redundant Reset Factor Clear Register

This register clears the reset flags of the RESFR.

**Access:** This register is a write-only register that can be written in 32-bit units.

**Address:** FFF8 0868$_H$

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | RESFCR10 | RESFCR9 | RESFCR8 | RESFCR7 | RESFCR6 | RESFCR5 | RESFCR4 | RESFCR3 | RESFCR2 | RESFCR1 | RESFCR0 |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | W | W | W | W | W | W | W | W | W | W | W |

**Table 9.8 RESFCR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When writing, write “0”. |
| 10 | RESFCR10 | Reset flag clear by DeepSTOP mode<br>0: Do not clear flag<br>1: Clear flag |
| 9 | RESFCR9 | Power-up reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 8 | RESFCR8 | External reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 7 | RESFCR7 | CVM reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 6 | RESFCR6 | LVI reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 5 | RESFCR5 | CLMA2 reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 4 | RESFCR4 | CLMA1 reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 3 | RESFCR3 | CLMA0 reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 2 | RESFCR2 | WDTA1 reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 1 | RESFCR1 | WDTA0 reset flag clear<br>0: Do not clear flag<br>1: Clear flag |
| 0 | RESFCR0 | Software reset flag clear<br>0: Do not clear flag<br>1: Clear flag |

## 9.3.3 Details of Software Reset Control Registers

### 9.3.3.1 SWRESA — Software Reset Register

This register is used to generate a software reset SWRES. The correct write sequence using the PROTCMD0 register is required in order to update this register.

For details, see **Section 5, Write-Protected Registers**.

**Access:** This register is a write-only register that can be written in 32-bit units.

**Address:** FFF8 0A04$_H$

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWRESA |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 9.9 SWRESA Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write “0”. |
| 0 | SWRESA | Software reset trigger<br>0: No Software reset trigger is generated.<br>1: Software reset trigger is generated. |

## 9.3.4 Details of Cyclic RUN Mode Reset Vector Address Register

### 9.3.4.1 CYCRBASE — Cyclic RUN mode RBASE Register

This register is used to specify the reset vector address (RBASE) of the CPU, when the CPU returns to Cyclic RUN mode from DeepSTOP mode. The correct write sequence using the PROTCMD0 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

Update of this register is only permitted when the chip is in RUN mode. Do not change the value in Cyclic RUN mode.

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 $3600_H$

**Value after reset:** FEBF $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CYCRBASE[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CYCRBASE[15:9] | | | | | | | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R |

**Table 9.10 CYCRBASE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | CYCRBASE | Reset vector base address (RBASE) in Cyclic RUN mode.<br>Specifies the RBASE value in Cyclic RUN mode.<br>The default value is set to the start address of the Retention RAM. |
| 8 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

## 9.4 Functional Description

### 9.4.1 Reset Flags

The reset factor register (RESF) and the redundant reset factor register (RESFR) provide reset flags for each reset source.

If a reset has occurred, the corresponding flag is set. According to this, the source of the reset is evaluated.

RESF and RESFR are initialized by a power-up reset PURES (POCRES or $\overline{\text{DBRES}}$) (though bits RESF9 and RESFR9 are set to 1 after initialization). In addition, flags in RESF and RESFR can be cleared by the reset factor clear register (RESFC) and the redundant reset factor clear register (RESFRC).

Each reset source can set the corresponding flag independently from other reset sources.

### 9.4.2 Power-On-Clear (POC) Reset

The power-on-clear circuit (POC) constantly compares the power supply voltage REGVCC with the internal reference voltage $V_{POC}$. It ensures that the microcontroller only operates as long as the power supply exceeds a certain level.

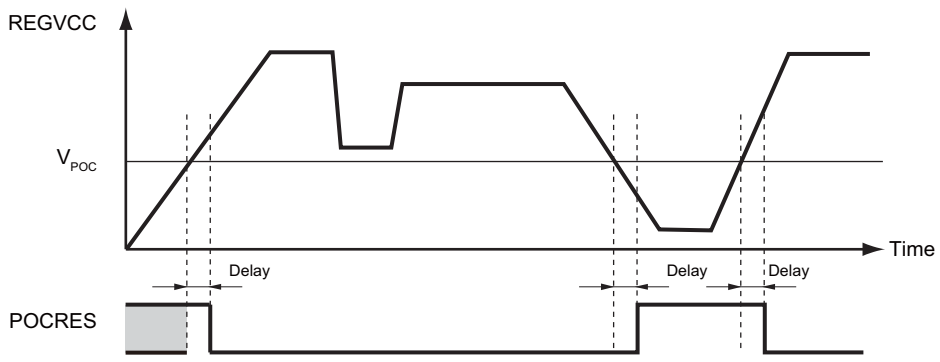
If REGVCC falls below the internal reference voltage (REGVCC < $V_{POC}$), the internal reset signal POCRES and a power-up reset PURES are generated.

For details on the specification of the internal voltage reference level $V_{POC}$, see **Section 40, Electrical Characteristics**.

The reset factor register (RESF) and the redundant reset factor register (RESFR) are cleared by the power-on clear reset. RESF9 and RESFR9 are set to 1 after initialization.

The power-on-clear function holds the microcontroller in reset state as long as the power supply voltage does not exceed the threshold level $V_{POC}$.

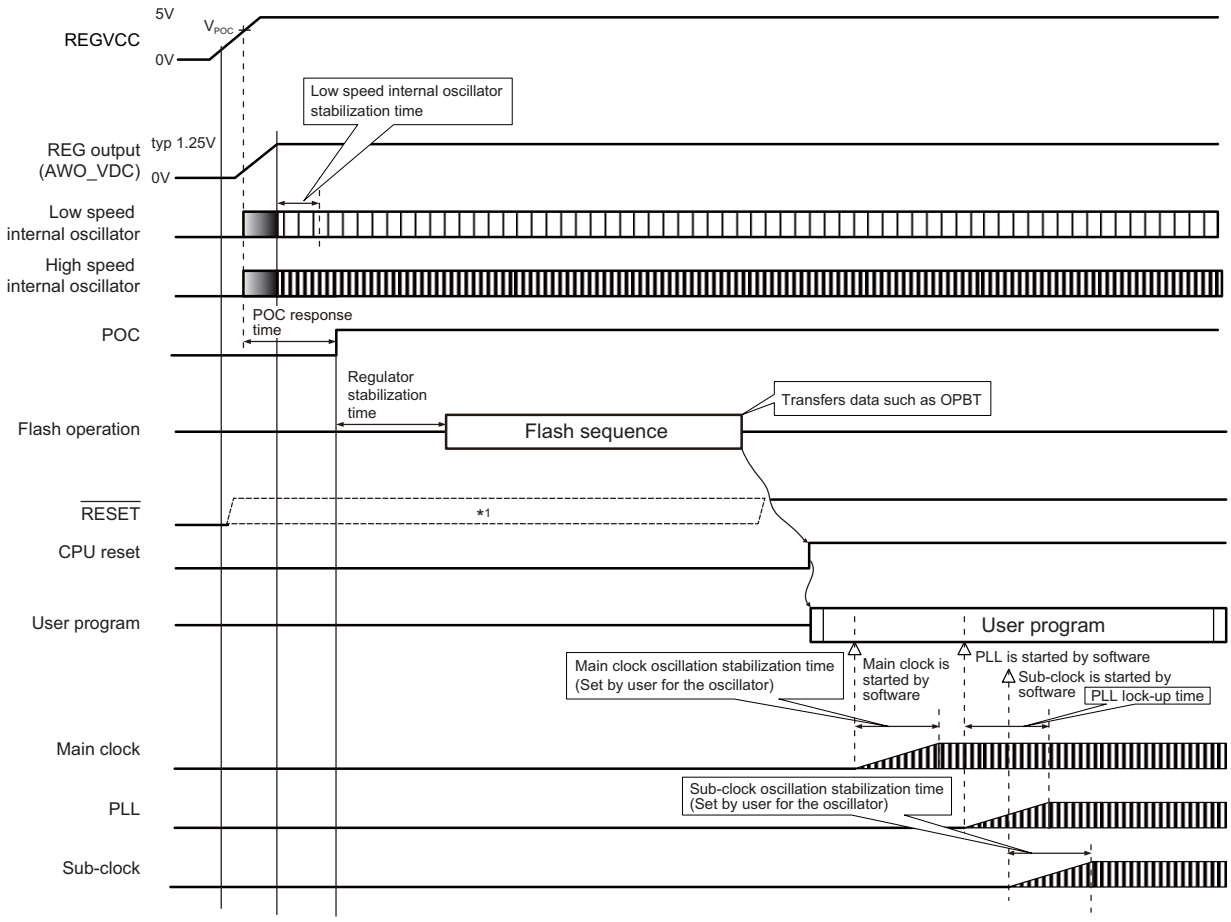
The following figure illustrates the timing of a POCRES.




**Note:** Delay:
When the REGVCC exceeds the $V_{POC}$, the POCRES is released after certain delay. When the REGVCC falls below the $V_{POC}$, the POCRES is asserted after certain delay.
See **Section 40, Electrical Characteristics** for the delay time.

**Figure 9.3 POC Reset Timing**

### (1) Overview of CPU system startup after power-on-clear




Note 1. The timing of releasing the $\overline{\text{RESET}}$ pin depends on the external circuit of the microcontroller.
If the $\overline{\text{RESET}}$ pin is low level, the values after reset of both RESF.RESF8 and RESFR.RESFR8 are set to 1. If the $\overline{\text{RESET}}$ pin is high level, the values after reset of both RESF.RESF8 and RESFR.RESFR8 are set to 0.

**Figure 9.4 When $\overline{\text{RESET}}$ is Released before the Flash Sequence is Completed**

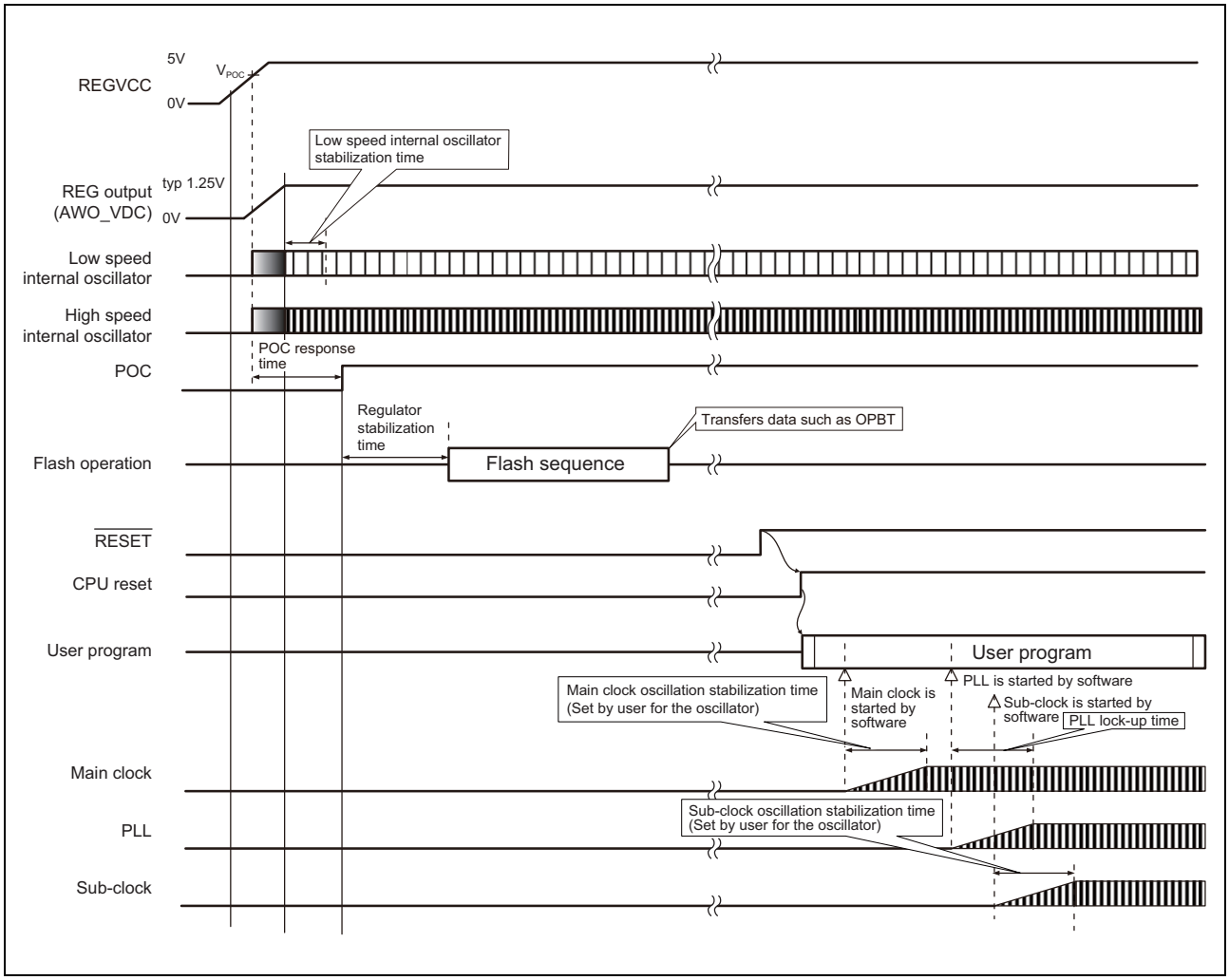



**Figure 9.5 When $\overline{\text{RESET}}$ is Released after the Flash Sequence is Completed**

### 9.4.3 Low-Voltage Indicator (LVI) Reset

The low-voltage indicator (LVI) constantly compares the power supply voltage REGVCC with the LVI internal reference voltage $V_{LVI}$.

When setting the LVI detection voltage and releasing the LVIRESMK, if REGVCC falls below the internal reference voltage (REGVCC < $V_{LVI}$), the internal reset signal $\overline{\text{LVIRES}}$ is generated.

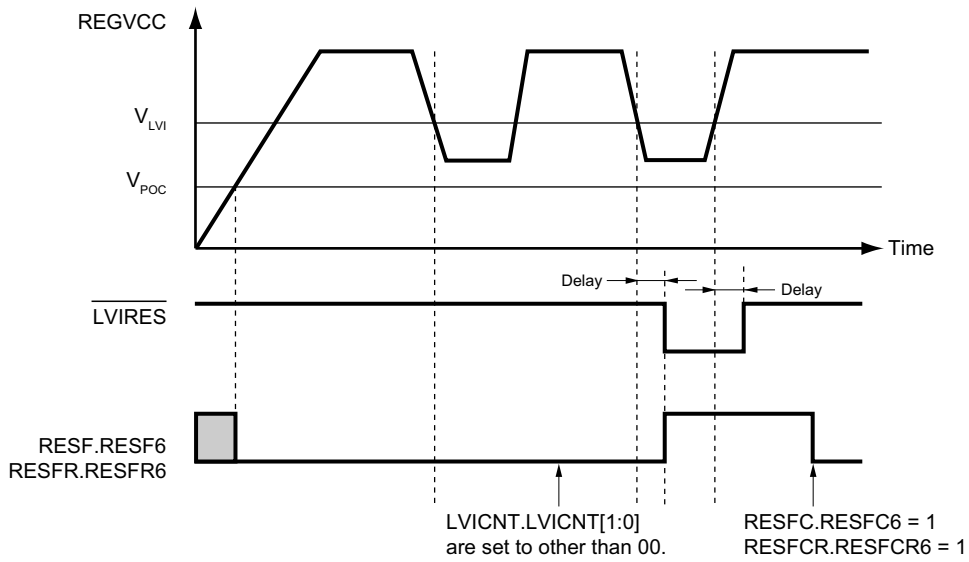
Additionally, the $\overline{\text{LVIRES}}$ flags (bits RESF.RESF6 and RESFR.RESFR6) are set.

After that, even if REGVCC exceeds $V_{LVI}$, bits RESF.RESF6 and RESFR.RESFR6 are not cleared automatically. They are cleared as described below.

- Setting the RESFC.RESFC6 bit to 1 clears the RESF.RESF6 bit.
  Setting the RESFCR.RESFCR6 bit to 1 clears the RESFR.RESFR6 bit.
- Power-up reset PURES (POCRES or $\overline{\text{DBRES}}$)

For details on the LVI functions, see **Section 11, Supply Voltage Monitor**.

The following figure illustrates the timing of a $\overline{\text{LVIRES}}$ and bits RESF.RESF6 and RESFR.RESFR6.




**Note:** Delay:
When REGVCC falls below $V_{LVI}$, $\overline{\text{LVIRES}}$ is asserted and the VLVI reset flags (RESF.RESF6 and RESFR.RESFR6) are set after a certain delay. After that, when REGVCC exceeds $V_{LVI}$, $\overline{\text{LVIRES}}$ is released after a certain delay.
See **Section 40, Electrical Characteristics** for the delay time.

**Figure 9.6 LVI Reset Timing**

### 9.4.4 Core Voltage Monitor (CVM) Reset

Core voltage monitor is used to monitor the core voltage inside the microcontroller.

The reset $\overline{\text{CVMRES}}$ is generated if the core voltage is not in the specified voltage range while CVM is enabled. Moreover, the $\overline{\text{CVMRES}}$ flags (RESF.RESF7 and RESFR.RESFR7) are set.

After that, the RESF.RESF7 and RESFR.RESFR7 bits are not automatically cleared even if the core voltage returns to the specified voltage range. The RESF.RESF7 and RESFR.RESFR7 bits are cleared as described below.

- Setting the RESFC.RESFC7 bit to 1 clears the RESF.RESF7 bit.
  Setting the RESFCR.RESFCR7 bit to 1 clears the RESFR.RESFR7 bit.
- Power-up reset PURES (POCRES or $\overline{\text{DBRES}}$)

If the CVM detects an abnormal high voltage, the power supply to the Isolated area (ISO area) is switched off. Once $\overline{\text{CVMRES}}$ is generated upon high voltage detection, the microcontroller stays in the reset state. To cancel this state, it is mandatory to use the external reset ($\overline{\text{RESET}}$) input. Release the external reset ($\overline{\text{RESET}}$) after the voltage level becomes lower than the high detection voltage.

For details on the CVM function, see **Section 11, Supply Voltage Monitor**.

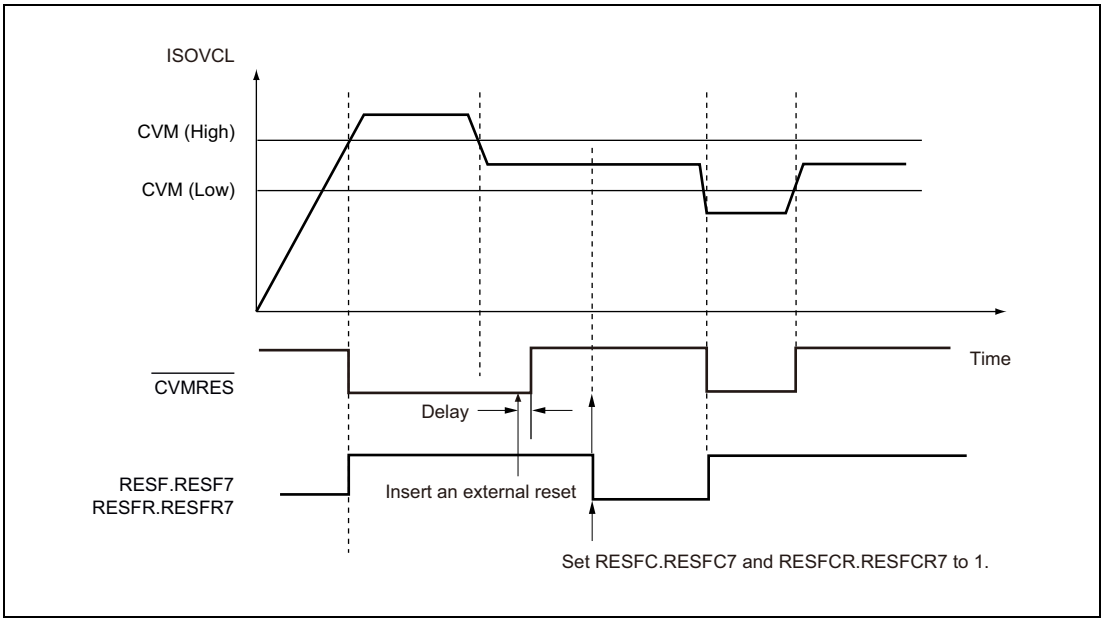



**Figure 9.7 CVM Reset Timing**

### 9.4.5 External Reset ($\overline{\text{RESET}}$)

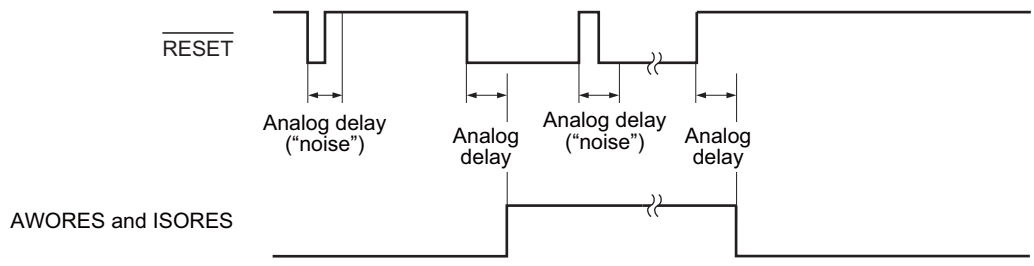
When a low level input is applied to the $\overline{\text{RESET}}$ pin, a reset is asserted and the RESF.RESF8 and RESFR.RESFR8 bits are set.

After that, bits RESF.RESF8 and RESFR.RESFR8 are not cleared automatically, even if the low-level input to the $\overline{\text{RESET}}$ pin is released. Bits RESF.RESF8 and RESFR.RESFR8 are cleared as described below.

- Setting the RESFC.RESFC8 bit to 1 clears the RESF.RESF8 bit.
  Setting the RESFCR.RESFCR8 bit to 1 clears the RESFR.RESFR8 bit.
- Power-up reset PURES (POCRES or $\overline{\text{DBRES}}$)

The $\overline{\text{RESET}}$ pin includes an analog noise filter to prevent erroneous resets due to noise.

The following figure shows the timing when AWORES and ISORES are generated by an external reset. This figure also shows the effect of the noise filter.




**Note:** The analog delay is caused by the analog filter. The filter regards pulses up to a certain width as noise and suppresses them.
For the minimum $\overline{\text{RESET}}$ pulse width, see **Section 40, Electrical Characteristics**.

**Figure 9.8 External Reset ($\overline{\text{RESET}}$)**

### 9.4.6 Watchdog Timer (WDTA) Reset

The watchdog timers can be configured to generate a reset if the overflow time is exceeded. After a watchdog timer reset is asserted, the corresponding watchdog timer reset flags (the RESF.RESF1 and RESFR.RESFR1 bits for WDTA0RES, and the RESF.RESF2 and RESFR.RESFR2 bits for WDTA1RES) are set.

After that, bits RESF.RESF1 and RESFR.RESFR1 (or bits RESF.RESF2 and RESFR.RESFR2) are not cleared automatically, even if WDTA0RES (or WDTA1RES) is released.

Bits RESF.RESF1 and RESFR.RESFR1, and bits RESF.RESF2 and RESFR.RESFR2 are cleared as described below.

- WDTA0RES:
  Setting the RESFC.RESFC1 bit to 1 clears the RESF.RESF1 bit.
  Setting the RESFCR.RESFCR1 bit to 1 clears the RESFR.RESFR1 bit.
- WDTA1RES:
  Setting the RESFC.RESFC2 bit to 1 clears the RESF.RESF2 bit.
  Setting the RESFCR.RESFCR2 bit to 1 clears the RESFR.RESFR2 bit.
- Power-up reset PURES (POCRES or $\overline{\text{DBRES}}$)

### 9.4.7 Software Reset

The software reset SWRES can be asserted by setting SWRESA.SWRESA to 1.

SWRES sets the reset flag RESF.RESF0 and the RESFR.RESFR0 bit.

RESF.RESF0 and RESFR.RESFR0 are not cleared automatically. RESF.RESF0 and RESFR.RESFR0 are cleared as described below.

- Setting the RESFC.RESFC0 bit to 1 clears the RESF.RESF0 bit.
  Setting the RESFCR.RESFCR0 bit to 1 clears the RESFR.RESFR0 bit.
- Power-up reset PURES (POCRES or $\overline{\text{DBRES}}$)

### 9.4.8 Clock Monitor (CLMA) Reset

The clock monitors can generate the following resets:

- $\overline{\text{CLMA0RES}}$, if a frequency abnormality in HS IntOSC is detected
- $\overline{\text{CLMA1RES}}$, if a frequency abnormality in MainOSC is detected
- $\overline{\text{CLMA2RES}}$, if a frequency abnormality in PLL is detected

When the Clock Monitor detects frequency abnormality of the respective clocks, resets $\overline{\text{CLMA0RES}}$, $\overline{\text{CLMA1RES}}$, and $\overline{\text{CLMA2RES}}$ are generated.

In addition, flags $\overline{\text{CLMA0RES}}$, $\overline{\text{CLMA1RES}}$, and $\overline{\text{CLMA2RES}}$ (RESF.RESF3, RESFR.RESFR3, RESF.RESF4, RESFR.RESFR4, RESF.RESF5, and RESFR.RESFR5) are set.

These flags are not cleared automatically. They are cleared as described below.

- $\overline{\text{CLMA0RES}}$:
  Setting the RESFC.RESFC3 bit to 1 clears the RESF.RESF3 bit.
  Setting the RESFCR.RESFCR3 bit to 1 clears the RESFR.RESFR3 bit.
- $\overline{\text{CLMA1RES}}$:
  Setting the RESFC.RESFC4 bit to 1 clears the RESF.RESF4 bit.
  Setting the RESFCR.RESFCR4 bit to 1 clears the RESFR.RESFR4 bit.
- $\overline{\text{CLMA2RES}}$:
  Setting the RESFC.RESFC5 bit to 1 clears the RESF.RESF5 bit.
  Setting the RESFCR.RESFCR5 bit to 1 clears the RESFR.RESFR5 bit.
- Power-up reset PURES (POCRES or $\overline{\text{DBRES}}$)

### 9.4.9 Debugger Reset

Debugger reset ($\overline{\text{DBRES}}$) is generated via a debugger command. $\overline{\text{DBRES}}$ activates PURES, and therefore operates in the same way as the power-on-clear reset POCRES:

- The clock generators are reset and stop operating. The clock generators should be restarted after release from the reset state.
- The reset factor register RESF and the redundant reset factor register RESFR are cleared (Bits RESF9 and RESFR9 are set to 1 after initialization.).

### 9.4.10 Reset Vector Address of CPU

The default value of the reset vector base address (RBASE) of CPU is set to 0000 $0000_H$ at shipment. In Cyclic RUN mode, the reset vector base address is automatically switched to the address specified by the CYCRBASE register. After returning to the RUN mode, the reset vector base address automatically switches back to the default address.

For details about the RBASE register, see **Section 3, CPU System**.

# Section 10 Power Supply Circuit

This section describes the power supply and power domains of the RH850/F1K.

## 10.1 Function

The internal circuits are separated into two independent power domains, the Always-On area (AWO area) and the Isolated area (ISO area).

The power supply of the Always-On area is always on in all operating modes and stand-by modes.

The power supply of the Isolated area can be turned off to reduce the overall power consumption depending on the type of stand-by mode.

For each power domain, a dedicated on-chip voltage regulator generates the internal supply voltage.

For operation of the device, the following voltages are required:

- Power supply voltage REGVCC for the on-chip voltage regulators. The output voltage of the voltage regulators is supplied to the digital circuits in each power domain.
- Power supply voltage EVCC for the I/O ports.
- Power supply voltages A0VREF and A1VREF*1 for the A/D converters and the separated I/O ports.

**Note 1.** Only available for 144 pin, and 176 pin devices.

### 10.1.1 Power Supply Pins

The table below lists all power supply pins and what they are used for.

**Table 10.1 Power Supply Pins**

| Power Supply | Power Supply Pins | Power Supply for |
|---|---|---|
| Power supply for internal circuits | REGVCC | • On-chip voltage regulators for the Always-On area and Isolated area<br>• Port group IP0*2<br>• MainOSC<br>• SubOSC*2<br>• POC / LVI |
| | AWOVCL*1 | |
| | AWOVSS | |
| | ISOVCL*1 | |
| | ISOVSS | |
| Power supply for I/O port | EVCC | • $\overline{\text{RESET}}$<br>• FLMD0<br>(176 pin devices)<br>• Port groups JP0, P0, P1, P2, P8, P9, P10, P11, P12, P18, P20<br>(144 pin devices)<br>• Port groups JP0, P0, P1, P8, P9, P10, P11, P12, P18, P20<br>(100 pin devices)<br>• Port groups JP0, P0, P8, P9, P10, P11 |
| | EVSS | |
| Power supply for A/D converters | A0VREF | • Analog circuits of ADCA0, port group AP0 |
| | A0VSS | |
| | A1VREF | (144 pin and 176 pin devices)<br>• Analog circuits of ADCA1, port group AP1 |
| | A1VSS | |

**Note:** See **Section 40, Electrical Characteristics** for the voltage range of each power supply.

Note 1. Pin to connect a stabilization capacitor for on-chip voltage regulator.

Note 2. Supported only by the 144 pin and 176 pin devices.

### 10.1.2 Block Diagram of Power Domains

The figure below shows the overview of power supply circuit.

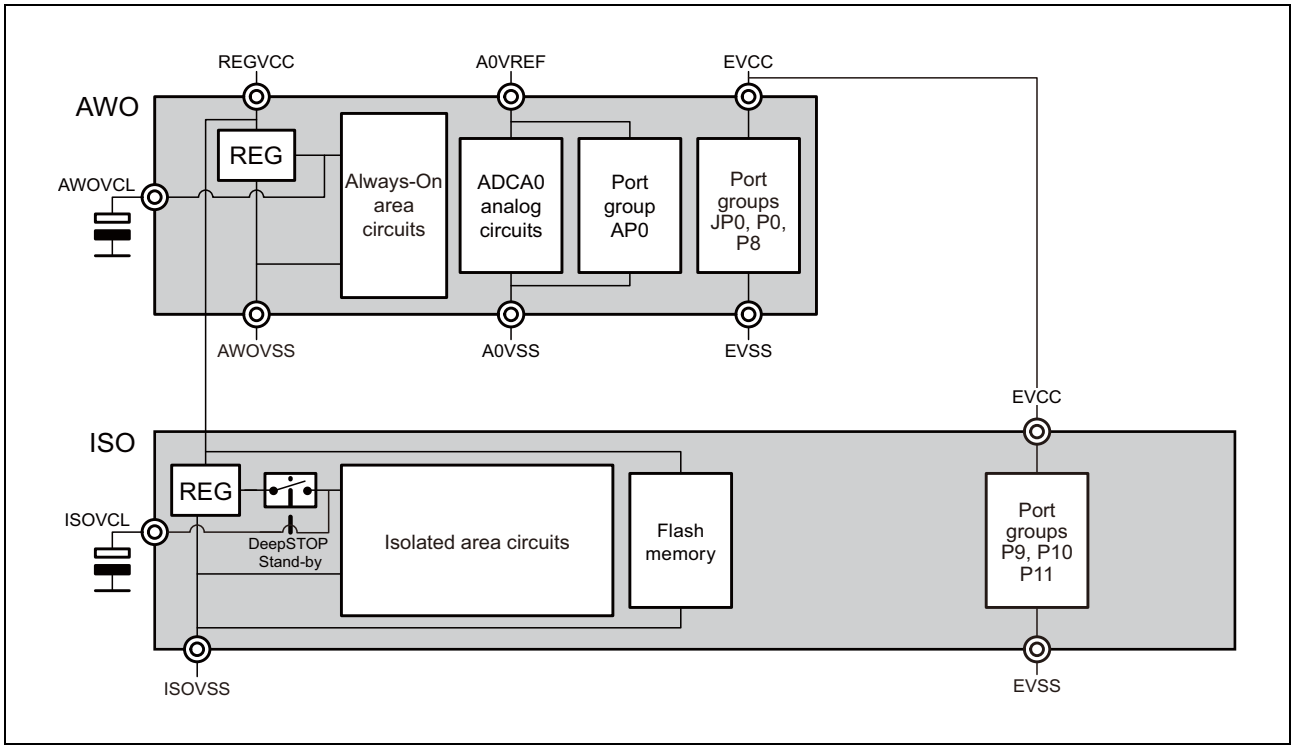



**Figure 10.1 Overview of Power Supply Circuit (100 pin)**

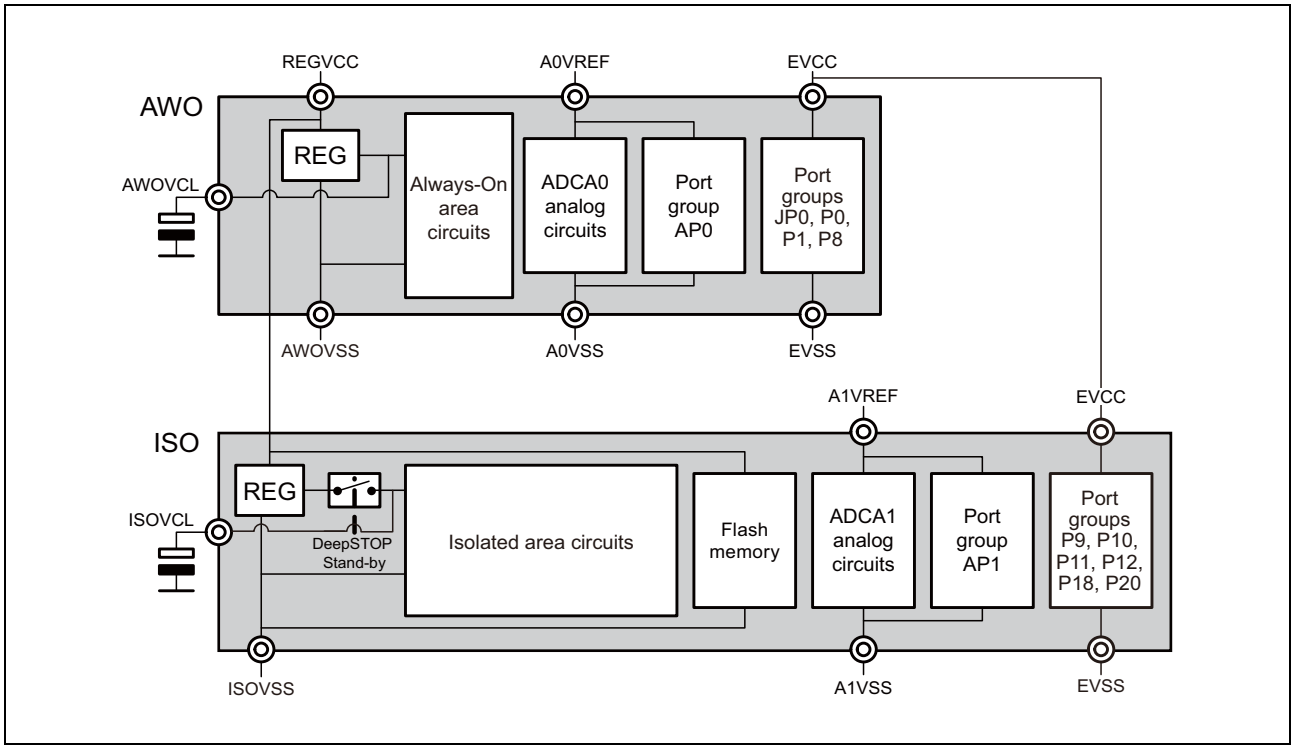



**Figure 10.2 Overview of Power Supply Circuit (144 pin)**

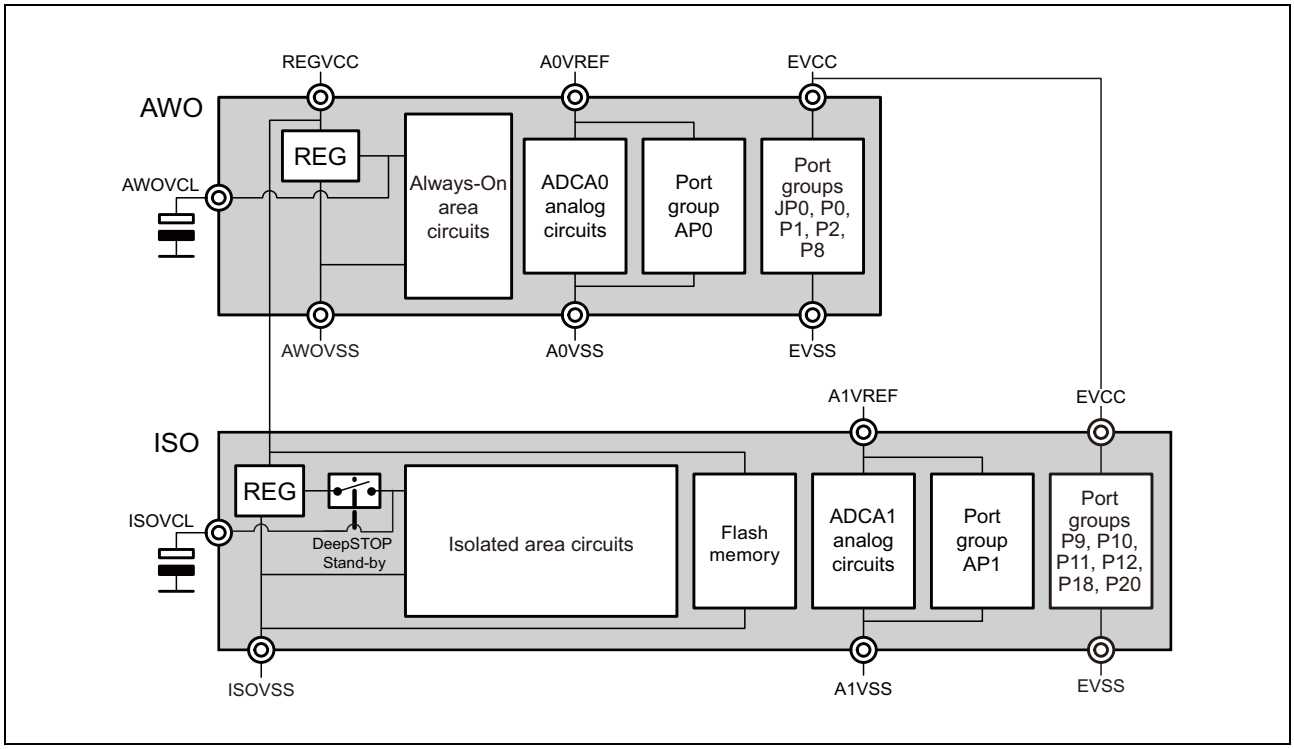



**Figure 10.3 Overview of Power Supply Circuit (176 pin)**

### 10.1.3 Power Domains Arrangement

The table below lists the microcontroller functional modules for each power domain.

**Table 10.2 Functional Modules and Power Domain**

| Power Domain | Functions |
|---|---|
| Always-On area (AWO area) | • STBC, Reset controller<br>• Retention RAM<br>• MainOSC, SubOSC, LS IntOSC, HS IntOSC, CLMA0, CLMA1<br>• WDTA0, RTCAn, TAUJ0, ADCA0, LPS<br>• Port groups JP0, P0, P1, P2, P8, AP0, IP0 |
| Isolated area (ISO area) | • CPU subsystem<br>• Code flash, Data flash, Local RAM<br>• PLL, CLMA2<br>• WDAT1, DCRAn, TAUDn, TAUBn, TAUJ1, OSTMn, PWM-Diag, CSIGn, CSIHn, RSCANn, RLIN2m, RLIN3n, RIICn, ADCA1, Motor Control, ENCAn, KRn<br>• Port groups P9, P10, P11, P12, P18, P20, AP1 |

# Section 11 Supply Voltage Monitor

This section explains in general about the supply voltage monitor.

The first part in this section describes the supply voltage monitor function, and the ensuing sections describe the registers.

This supply voltage monitor is for detecting and control of a power supply failure. However, the supply voltage monitor does not detect all of the possible failures.

Therefore, a power supply monitoring with an external device is required for the following terminals, in case that the customer's system requires an appropriate failure detection and control for Functional Safety.

- REGVCC
- EVCC
- A0VREF
- A1VREF
- AWOVCL
- ISOVCL

For the detail of required power supply specification for power supply monitoring with external device, see **Section 40.8, Power Management Characteristics**.

## 11.1 Overview

### 11.1.1 Functional Overview

The supply voltage monitor continuously monitors multiple external and internal supply voltages in order to ensure that the device operates with a supply voltage within the specified range. If the voltage drops below the reference voltage or comparison voltage, an interrupt request signal or internal reset signal is generated. The following table lists the supply voltage monitor functions.

**Table 11.1 Supply Voltage Monitor Functions**

| Function Name | Monitor Voltage | Signal Generated when Voltage Drops below Certain Level |
|---|---|---|
| Power-On Clear (POC) | REGVCC | Internal reset signal |
| Low-Voltage Indicator Circuit (LVI) | REGVCC | Internal reset signal, interrupt request signal |
| Core Voltage Monitor (CVM) | Isolated area voltage | Internal reset signal |
| RAM Retention Voltage Indicator (VLVI) | REGVCC | — |

**NOTES**

1. The RAM Retention Voltage Indicator sets the very-low voltage detection flag (VLVF) when the voltage drops below the RAM retention voltage.
2. When the internal core voltage monitor (CVM) is used for customer's system as the functional safety measure, the voltage of the Always-On area (AWO area) shall be monitored by the external voltage monitor.

### 11.1.2 Power-On Clear (POC)

The POC continuously monitors the external power supply voltage REGVCC. This ensures that the microcontroller only operates at or above power-on-clear detection voltage ($V_{POC}$).

If REGVCC falls below the POC detection voltage (REGVCC < $V_{POC}$), the internal reset signal (POCRES) is generated.

For details, see **Section 9.4.2, Power-On-Clear (POC) Reset**.

### 11.1.3 Low Voltage Indicator Circuit (LVI)

The LVI continuously compares the external power supply voltage REGVCC with the LVI reference voltage $V_{LVI}$.

If REGVCC falls below the reference voltage (REGVCC < $V_{LVI}$), an internal reset signal or interrupt request signal is generated.

#### 11.1.3.1 LVI Reference Voltage

The LVI reference voltage $V_{LVI}$ can be selected from three different levels by LVICNT.LVICNT[1:0].

If LVICNT.LVICNT[1:0] is set to $00_B$, the LVI is disabled.

For the specification of the reference voltage level $V_{LVI}$, see **Section 11.2.2.1, LVICNT — LVI Control Register**.

#### 11.1.3.2 LVI Reset (LVIRES)

When the LVI detection voltage is set and LVIRESMK is cleared, if REGVCC falls below the reference voltage (REGVCC < $V_{LVI}$), the internal reset signal $\overline{\text{LVIRES}}$ is generated.

For the specification of $\overline{\text{LVIRES}}$ generation, see **Section 9.4.3, Low-Voltage Indicator (LVI) Reset**.

### 11.1.3.3 LVI Interrupt (INTLVIL/INTLVIH)

After the LVI detection voltage is set to LVICNT.LVICNT[1:0] and LVICNT.LVIRESMK is set to 1, if REGVCC falls below the reference voltage (REGVCC (MIN) < $V_{LVI}$), the LVI interrupt INTLVIL is generated.

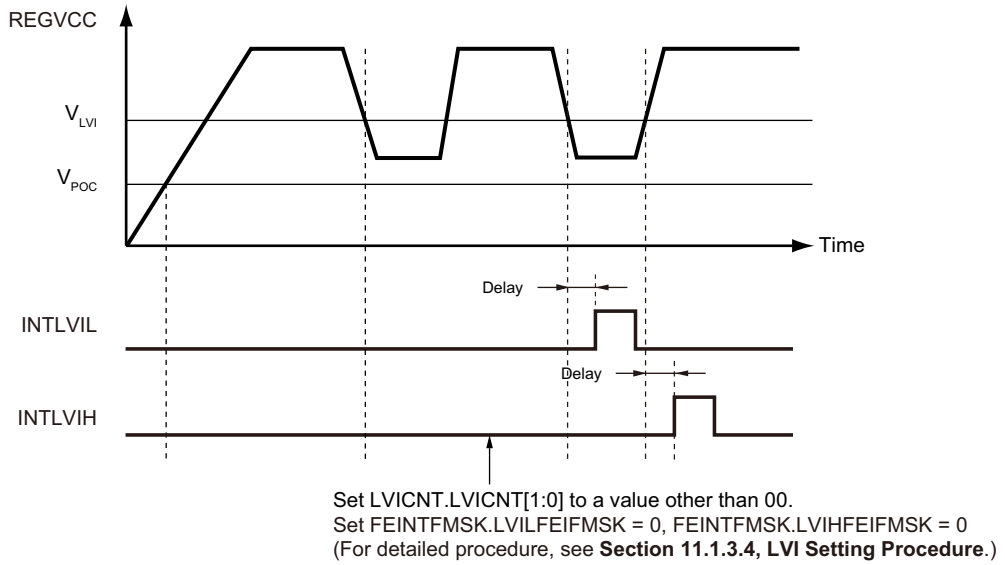
To use the LVI as an interrupt source, the INTLVIL interrupt must be unmasked.

INTLVIL interrupt can be used as wake-up source from all of standby modes. For details, see Section 14, Stand-By Controller (STBC).

After the LVI detection voltage is set to LVICNT.LVICNT[1:0] and LVICNT.LVIRESMK is set to 1, if REGVCC exceeds the reference voltage (REGVCC (MIN > $V_{LVI}$), LVI interrupt INTLVIH is generated.

When LVI is used as an interrupt source, INTLVIH interrupt must be unmasked.

The following figure illustrates the timing of INTLVIL/INTLVIH.




**Note:** Delay:
When REGVCC falls below $V_{LVI}$, INTLVIL is generated after a certain delay.
After that, when REGVCC exceeds $V_{LVI}$, INTLVIH is generated after a certain delay.
See **Section 40, Electrical Characteristics** for delay time.

**Figure 11.1 INTLVIL/INTLVIH Generation Timing**

### 11.1.3.4 LVI Setting Procedure

The setting procedures for LVI are shown below.

**(1) Using LVI as the reset source**

(1) Mask LVI reset. (LVICNT.LVIRESMK = 1)*1

(2) Mask LVI interrupt. (FEINTFMSK.LVILFEIFMSK = 1, FEINTFMSK.LVIHFEIFMSK = 1)

(3) Set detection voltage and enable operation. (Set LVICNT.LVICNT[1:0])*1

(4) Insert ample wait time by software (see **Section 40, Electrical Characteristics**).

(5) Unmask LVI reset. (LVICNT.LVIRESMK = 0)*1

**(2) Using LVI as the interrupt source (FEINT)**

(1) Mask LVI reset. (LVICNT.LVIRESMK = 1)*1

(2) Mask LVI interrupt. (FEINTFMSK.LVILFEIFMSK = 1, FEINTFMSK.LVIHFEIFMSK = 1)

(3) Set detection voltage and enable operation. (Set LVICNT.LVICNT[1:0])*1

(4) Insert ample wait time by software (see **Section 40, Electrical Characteristics**).

(5) Unmask LVI interrupt. (FEINTFMSK.LVILFEIFMSK = 0, FEINTFMSK.LVIHFEIFMSK = 0)

**Note 1.** Follow the register protection sequence to set LVICNT register because it is a write-protected register. For details on the write-protected registers, see **Section 5, Write-Protected Registers**.

**CAUTION**

**If REGVCC is not stable around the LVI detection level ($V_{LVI}$), correct judgment of whether INTLVIH or INTLVIL interrupt processing should proceed may not be possible.**

**For example, if multiple interrupts consisting of both INTLVIH and INTLVIL occur during INTLVIL interrupt processing due to REGVCC being unstable, the software cannot detect which type of interrupt was generated last.**

**Consequently, if the last interrupt generated was an INTLVIL interrupt, regardless of REGVCC (min.) being greater than $V_{LVI}$, the software erroneously judges that REGVCC (min.) < $V_{LVI}$.**

**Accordingly, take measures such as programming the software so that LVI detection interrupt processing is completed before a next LVI interrupt. Also, consider control of REGVCC.**

### 11.1.3.5 Clock Supply to the LVI

The clock supply to the LVI is shown in the following table.

**Table 11.2 Clock Supply to the LVI**

| Unit Name | Unit Clock Name | Supply Clock Name |
|---|---|---|
| LVI | Register access clock | CPUCLK4 |

### 11.1.4 Core Voltage Monitor (CVM)

The core voltage monitor (CVM) monitors the Isolated area voltage (referred to as “core voltage” below) in the microcontroller.

If the regulator output voltage is outside of the specified range, the internal reset signal ($\overline{\text{CVMRES}}$) is generated.

If the CVM detects an abnormal high voltage, the power supply to the Isolated area is switched off in addition to a reset being generated.

When operation shifts to diagnostic mode (DIAG mode), the CVM enters the abnormal core voltage detection state. An abnormal core voltage detected state can be intentionally created by using the DIAG mode so that the CVM abnormal voltage detected flag can be checked for failures.

**CAUTION**

**The CVM cannot detect drifts in the voltage being supplied to the on-chip voltage regulator and Isolated area, or the increase or decrease of voltage.**

#### 11.1.4.1 CVM Reset ($\overline{\text{CVMRES}}$)

If the core voltage exceeds the specified level while high-voltage monitor is enabled (CVMDE.H_D_E = 1), then $\overline{\text{CVMRES}}$ is generated and the power supply to the Isolated area is stopped.

If the core voltage falls below the specified level while low voltage monitor is enabled (CVMDE.L_D_E = 1), $\overline{\text{CVMRES}}$ is generated.

For the specification of $\overline{\text{CVMRES}}$ generation, see **Section 9.4.4, Core Voltage Monitor (CVM) Reset**.

#### 11.1.4.2 CVM Setting

Use the option byte to enable the high-voltage monitor and the low-voltage monitor. For details, see **Section 37.9, Option Bytes**.

#### 11.1.4.3 Diagnostic (DIAG) Mode

This product supports diagnostic mode.

In diagnostic (DIAG) mode, whether the CVM abnormal voltage detection flag is set to 1 can be checked.

In diagnostic mode, $\overline{\text{CVMRES}}$ is not output.

The setting procedure for diagnostic mode is described below.

Set the registers according to this procedure. Otherwise the operation is not guaranteed.

(1) Set CVMDIAG.CVM_DIAG_MASK.*1

(2) Set CVMDIAG.CVM_DIAG.*1

(3) Wait for 12 µs.*2

(4) Read the CVMF register to confirm that the H_V_F and L_V_F bits are set to 1 (if these bits are 0, the CVM does not operate normally, requiring error handling).

(5) Clear CVMDIAG.CVM_DIAG.*1

(6) Clear the CVMF register.*1

(7) Read the CVMF register to confirm that the H_V_F and L_V_F bits are set to 0 (if these bits are 1, go back to step 5 again).

(8) Clear CVMDIAG.CVM_DIAG_MASK.*1

**Note 1.** Follow the register protection sequence to set CVMF and CVMDIAG registers because these are write-protected registers. For details, see **Section 5, Write-Protected Registers**.

**Note 2.** At least 50 µs must elapse after the following conditions are fulfilled before step (4) is started.

- Release from HALT state
- Release from STOP mode
- Release from the reset state, when a reset other than a reset due to the CVM is generated in RUN/HALT state
- The CPU clock is switched
- Operation of the MainOSC is started or stopped
- Operation of the PLL is started or stopped

#### 11.1.4.4 Clock Supply to the CVM

The clock supply to the CVM is shown in the following table.

**Table 11.3 Clock Supply to the CVM**

| Unit Name | Unit Clock Name | Supply Clock Name |
|---|---|---|
| CVM | Register access clock | CPUCLK4 |

### 11.1.5 RAM Retention Voltage Indicator (Very-Low-Voltage Detection Circuit, VLVI)

The very-low-voltage detection circuit (VLVI) is used to detect the RAM retention voltage, and continuously compares the power supply voltage REGVCC with the RAM retention voltage $V_{VLVI}$.

See **Section 40, Electrical Characteristics** for the specification of the RAM retention voltage level $V_{VLVI}$.

#### 11.1.5.1 Clock Supply to the VLVI

The clock supply to the VLVI is shown in the following table.

**Table 11.4 Clock Supply to the VLVI**

| Unit Name | Unit Clock Name | Supply Clock Name |
|---|---|---|
| VLVI | Register access clock | CPUCLK4 |

#### 11.1.5.2 Retention RAM Content Retention

If the power supply voltage REGVCC does not fall below $V_{VLVI}$, the content of the Retention RAM (RRAM) is retained. See **Section 9.1.1, Reset Sources** for retention during reset.

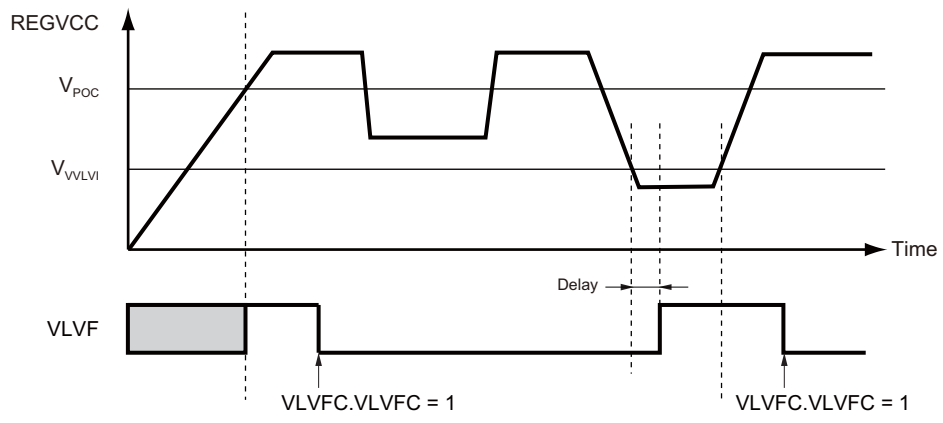
If REGVCC falls below $V_{VLVI}$, the RRAM content cannot be guaranteed. Thus the entire RRAM must be restored before continuing operation.

If REGVCC falls below the RAM retention voltage (REGVCC < $V_{VLVI}$), the VLVF.VLVF bit is set.

After that, even if REGVCC exceeds $V_{VLVI}$, the VLVF.VLVF bit is not cleared automatically. It is cleared by

- setting VLVFC.VLVFC bit to 1.

The following figure illustrates the timing of VLVF.




**Note:** Delay:
When REGVCC falls below $V_{VLVI}$, the VLVF bit is set to 1 after a certain delay.
See **Section 40, Electrical Characteristics** for delay time.

**Figure 11.2 VLVF Operation Timing**

### 11.1.6 Block Diagram

The block diagram of the supply voltage monitor is shown below.

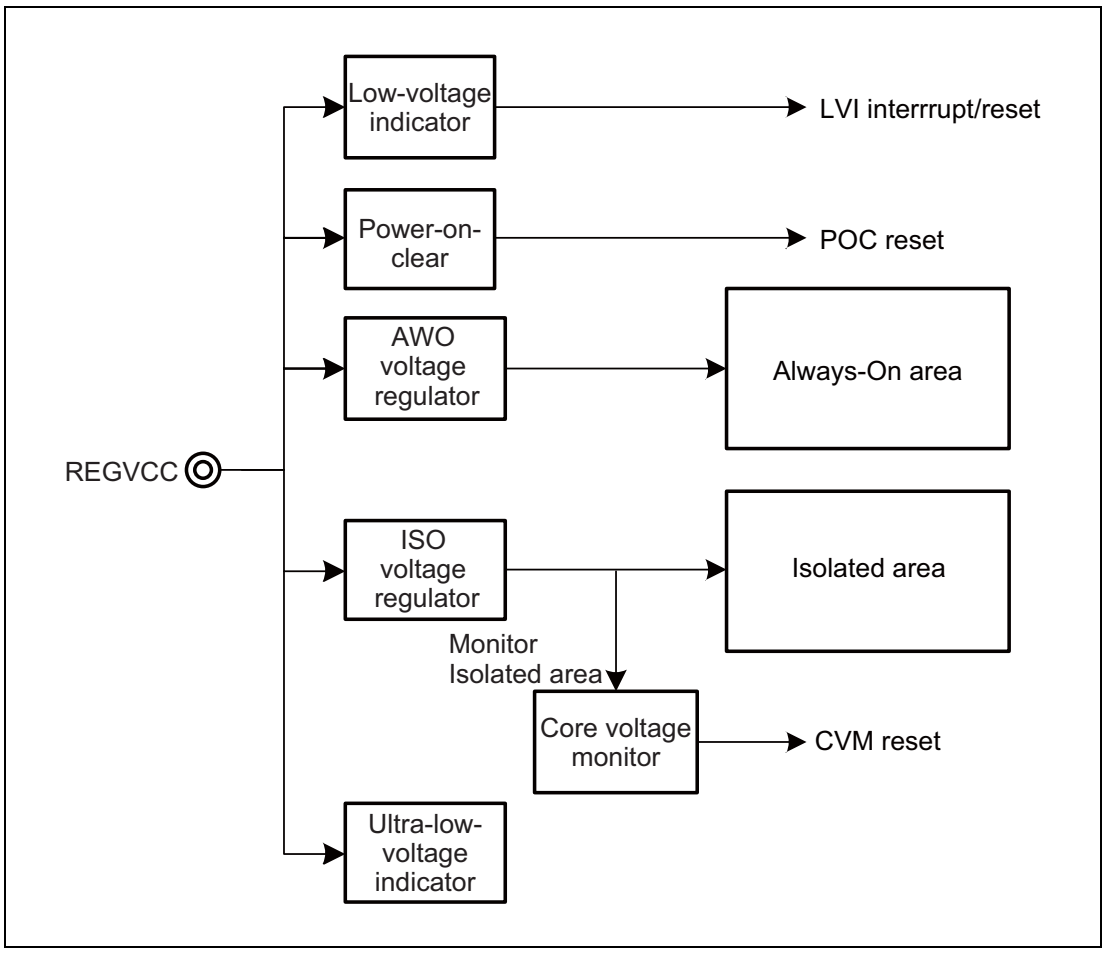



**Figure 11.3 Supply Voltage Monitor**

## 11.2 Registers

### 11.2.1 List of Registers

The following table lists the supply voltage monitor registers.

**Table 11.5 Power Supply Voltage Monitor Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| SVM | **Low-voltage indicator reset control register** | | |
| | LVI control register | LVICNT | FFF8 $0A00_H$ |
| | **Core voltage monitor control register** | | |
| | CVM factor register | CVMF | FFF8 $3100_H$ |
| | CVM detection enable register | CVMDE | FFF8 $3104_H$ |
| | CVM diagnostic mode setting register | CVMDIAG | FFF8 $3114_H$ |
| | **Very-low-voltage detection control register** | | |
| | Very-low-voltage detection register | VLVF | FFF8 $0980_H$ |
| | Very-low-voltage detection clear register | VLVFC | FFF8 $0988_H$ |

## 11.2.2 Low-Voltage Indicator Reset Control Registers

### 11.2.2.1 LVICNT — LVI Control Register

This register is used to control the Low-Voltage Indicator and to select the LVI detection level.

This register is initialized by power-up reset PURES.

The correct write sequence using the PROTCMD0 register is required in order to update this register For details, see **Section 5, Write-Protected Registers**.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 0A00$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | LVIRESMK | LVICNT[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 11.6 LVICNT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 | LVIRESMK | Mask LVI Reset<br>0: LVI reset is not masked<br>1: LVI reset is masked |
| 1, 0 | LVICNT[1:0] | Detection Level<br>0 0: LVI is ignored<br>0 1: 4.0+/-0.1 V (drop), 4.0+/-0.13 V (rise)<br>1 0: 3.7+/-0.1 V (drop), 3.7+/-0.13 V (rise)<br>1 1: 3.5+/-0.1 V (drop), 3.5+/-0.13 V (rise) |

**NOTE**

To use an LVI interrupt, LVI reset must be masked (LVIRESMK = 1) by LVIRESMK.

## 11.2.3 Core Voltage Monitor Control Registers

### 11.2.3.1 CVMF — CVM Factor Register

This register records the core voltage failure state generated after the last power-up reset PURES. The L_V_F bit and the H_V_F bit are set to 1 by hardware when the CVM detects core voltage failure.

If the L_V_F or H_V_F bit of this register is set to 1, that bit is not updated until it is initialized by a power-up reset PURES or by writing 0 to the CVMF.L_V_F or CVMF.H_V_F bit. However, it continuously monitors an error signal from the core voltage monitoring circuit in diagnostic mode.

The correct write sequence using the PROTCMDCVM register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 $3100_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | H_V_F | L_V_F |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 11.7 CVMF Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | H_V_F | High-Voltage Failure Detection of the Core Voltage by the CVM<br>Read access<br>0: No high-voltage failure state is detected<br>1: High-voltage failure state is detected<br>Write access<br>0: Clear the H_V_F bit<br>1: Invalid |
| 0 | L_V_F | Low-Voltage Failure Detection of the Core Voltage by the CVM<br>Read access<br>0: No low-voltage failure state is detected<br>1: Low-voltage failure state is detected<br>Write access<br>0: Clear the L_V_F bit<br>1: Invalid |

### 11.2.3.2 CVMDE — CVM Detection Enable Register

This register is used to indicate the voltage detection enabled or disabled state.
This register is initialized only by power-up reset PURES.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 $3104_H$

**Value after reset:** The value after reset depends on the option byte setting.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | H_D_E | L_D_E |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0/1*1 | 0/1*2 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. The setting of the option byte OPBT0.CVM_HD_EN is reflected.
Note 2. The setting of the option byte OPBT0.CVM_LD_EN is reflected.
For details on the option byte, see **Section 37.9, Option Bytes**.

**Table 11.8 CVMDE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1 | H_D_E | High-Voltage Monitor Enable<br>0: High-voltage detection is disabled.<br>1: High-voltage detection is enabled. |
| 0 | L_D_E | Low-Voltage Monitor Enable<br>0: Low-voltage detection is disabled.<br>1: Low-voltage detection is enabled. |

### 11.2.3.3 CVMDIAG — CVM Diagnostic Mode Setting Register

This register sets the CVM diagnostic mode.

This register is initialized only by power-up reset PURES.

For details on the register settings in diagnostic mode, see **Section 11.1.4.3, Diagnostic (DIAG) Mode**.

The correct write sequence using the PROTCMDCVM register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 $3114_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CVM_DIAG_MASK | CVM_DIAG |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 11.9 CVMDIAG Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | CVM_DIAG_MASK | $\overline{\text{CVMRES}}$ Mask Control<br>0: $\overline{\text{CVMRES}}$ is not masked.<br>1: $\overline{\text{CVMRES}}$ output is masked. |
| 0 | CVM_DIAG | CVM Diagnostic Mode Setting<br>0: Normal mode<br>1: Diagnostic mode |

## 11.2.4 Very-Low-Voltage Detection Control Registers

### 11.2.4.1 VLVF — Very-Low-Voltage Detection Register

The very-low-voltage detection register (VLVF) shows the state of the RAM retention voltage detection.

This register is set upon detection of a voltage below the RAM retention voltage ($V_{VLVI}$).

If VLVF is set, the Retention RAM content cannot be guaranteed.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 $0980_H$

**Value after power on:** 0000 $0001_H$
This register is not initialized by any reset source.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VLVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 11.10 VLVF Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | VLVF | Very-Low-Voltage Detection Flag<br>0: Very-low-voltage is not detected.<br>1: Very-low-voltage is detected.<br>**Note:** Very-low-voltage is the voltage status of REGVCC < RAM retention voltage ($V_{VLVI}$). For details, See **11.1.5.2, Retention RAM Content Retention** |

### 11.2.4.2 VLVFC — Very-Low-Voltage Detection Clear Register

This register clears the VLVF.VLVF bit.

**Access:** This register is a write-only register that can be written in 32-bit units.

**Address:** FFF8 0988$_H$

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VLVFC |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 11.11 VLVFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | VLVFC | Clear VLVF.VLVF bit.<br>0: Do not clear<br>1: Clear |

# Section 12 Clock Controller

This section explains in general about the clock controller.

The first part in this section describes the specific features of the clock controller of the RH850/F1K microcontrollers. The ensuing sections describe the clock oscillation circuit, clock selectors, and clock output function that make up the clock controller.

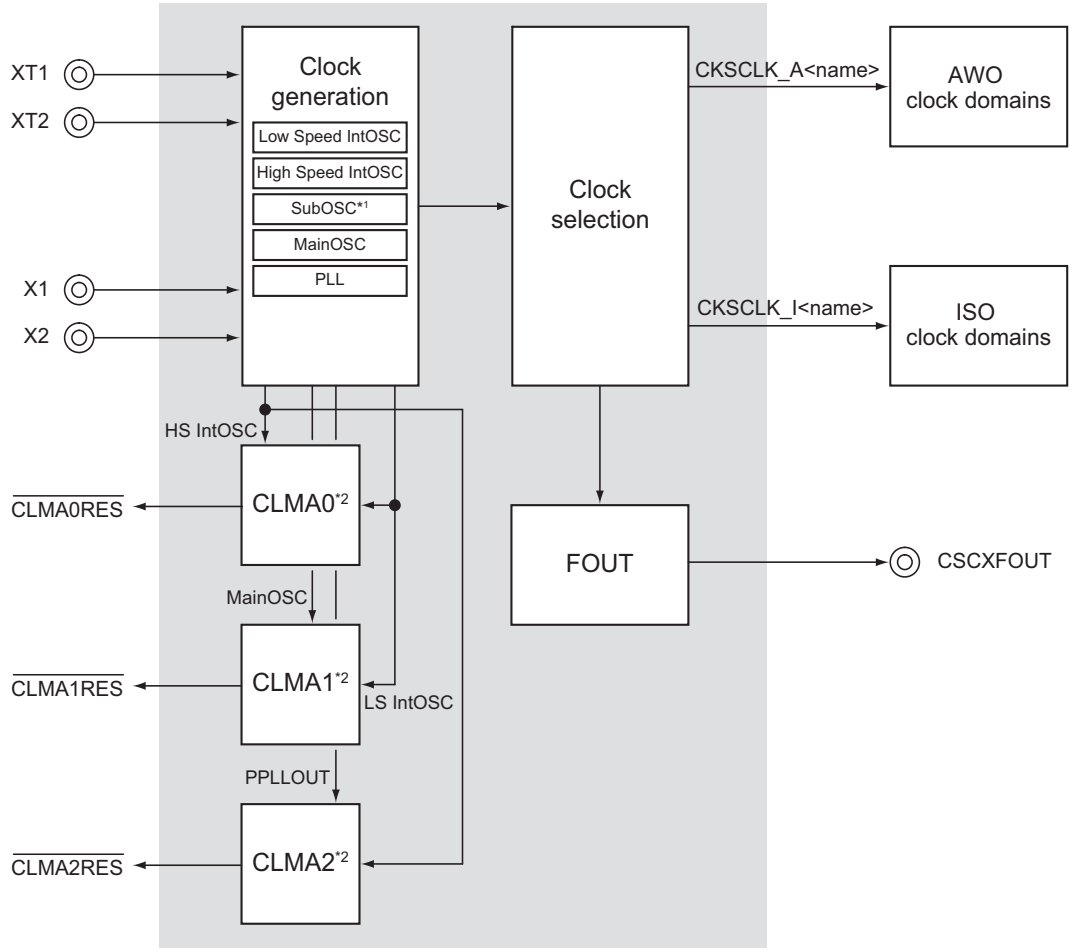
## 12.1 Features of Clock Controller of RH850/F1K

The clock controller of the RH850/F1K microcontrollers has the following features.

- Five on-chip clock oscillators
  - Main Oscillator (MainOSC) with an oscillation frequency of 16, 20, and 24 MHz
  - Sub Oscillator (SubOSC) with an oscillation frequency of 32.768 kHz*1
  - High Speed Internal Oscillator (HS IntOSC) with a nominal frequency of 8 MHz (Typ.)
  - Low Speed Internal Oscillator (LS IntOSC) with a nominal frequency of 240 kHz (Typ.)
  - PLL
- Fine management of clock supply to peripheral modules through clock domains
- On-chip clock monitor that detects clock anomalies when the Main Oscillator, High Speed Internal Oscillator, or PLL are in use. See Section 13, Clock Monitor (CLMA).
- Clock output (FOUT)

**Note 1.** The Sub Oscillator is supported only for the 144 pin and 176 pin products.

**Figure 12.1** shows the schematic diagram of the clock controller.




Note 1. Supported only by 144 pin and 176 pin products.
Note 2. See Section 13, Clock Monitor (CLMA) for details of clock monitors.

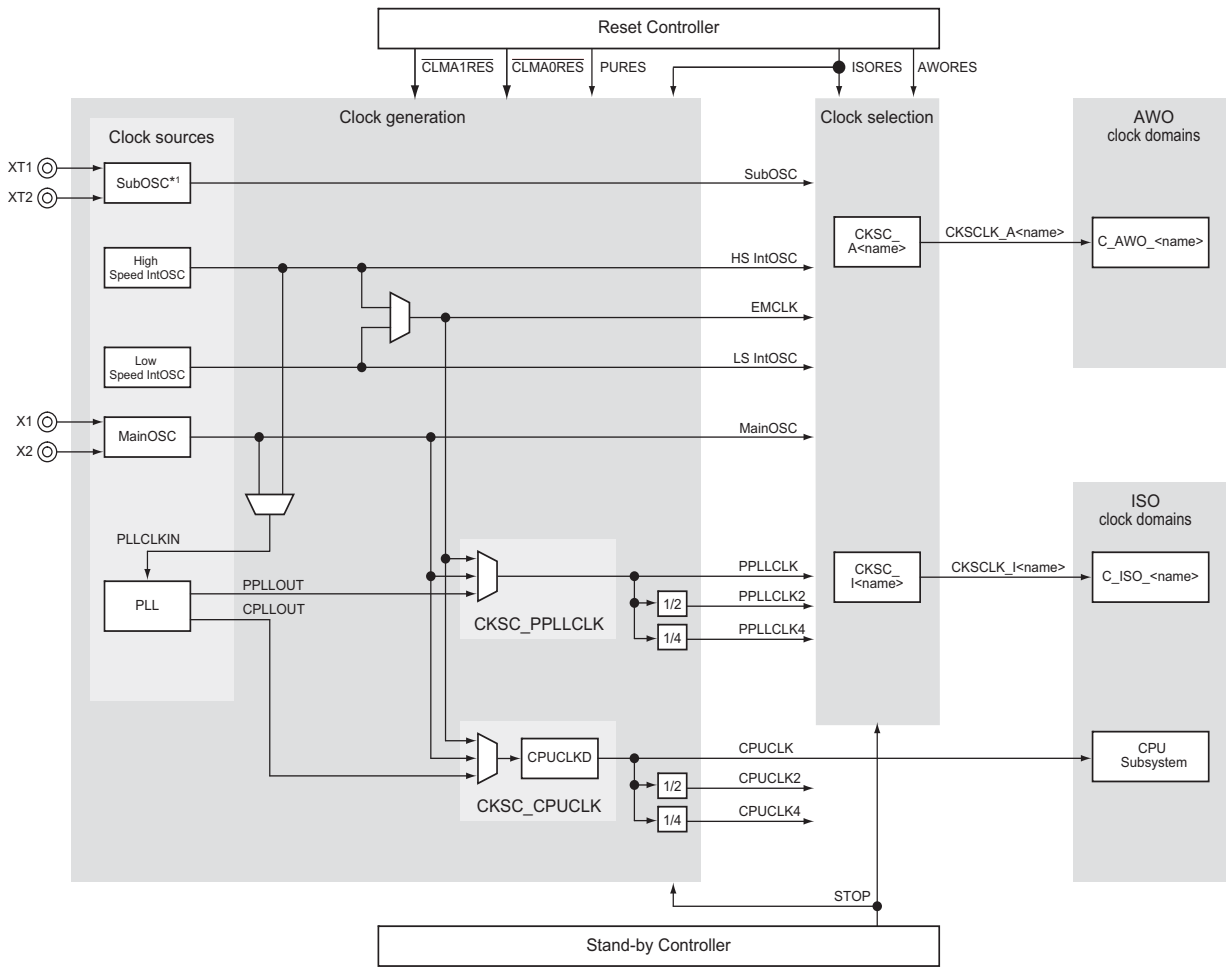
**Figure 12.1 Clock Controller Overview**

## 12.2 Configuration of Clock Controller

This section describes the configuration of the clock controller.

The clock controller is composed of clock oscillators and clock generation circuits that generate the clocks for the CPU and the peripheral modules, a clock selector for selecting the optimum clock, and clock domains for the CPU and the peripheral modules.

**Figure 12.2** shows the configuration of the clock controller.




Note 1. Supported only by 144 pin and 176 pin products.

**Figure 12.2 Clock Controller Configuration**

**NOTE**

**Clock domain and clock control register naming conventions**

The clock signals and their control registers, etc., described in this section are named according to the following naming conventions to reflect the power domain or clock domain to which they belong. The placeholder “<name>” is used to identify the target module in the clock domain:

- Clock domain names:
  - C_AWO_<name>: Always-On area*1 clock domain
  - C_ISO_<name>: Isolated area*1 clock domain
- Domain clock names:
  - CKSCLK_A<name>: Always-On area domain clock
  - CKSCLK_I<name>: Isolated area domain clock
- Clock selector names:
  - CKSC_A<name>: Always-On area clock selector
  - CKSC_I<name>: Isolated area clock selector
- Clock selector registers:
  - CKSC_A<name>S_CTL: Always-On area source clock selector register
  - CKSC_A<name>D_CTL: Always-On area source clock divider register
  - CKSC_I<name>S_CTL: Isolated area source clock selector register
  - CKSC_I<name>D_CTL: Isolated area source clock divider register

**Example**

The clock signal CKSCLK_AADCA (placeholder <name> = ADCA) is the clock supplied to the clock domain C_AWO_ADCA in the Always-On area. This clock is selected by the clock selector register CKSC_AADCAS_CTL.

**Note 1.** Always-On area and Isolated area refer to the power supply domains. Always-On area (AWO area) is an always-on power supply, and Isolated area (ISO area) is an a power supply that is switched on or off by the operation mode.
For details, see **Section 10, Power Supply Circuit**.

### 12.2.1 Clock Generation Circuits

Five clock oscillators are provided:

Four clock oscillators are located on the Always-On area (AWO area) and PLL is located on the Isolated area (ISO area).

#### Main Oscillator (MainOSC)

The MainOSC generates the main clock X.
Generation of the clock X requires the connection of an external resonator to X1 and X2.
The clock X is used as the reference clock for the PLL.

#### Sub Oscillator (SubOSC)

The SubOSC generates the sub-clock XT, which runs at a frequency of 32.768 kHz (Typ.). Generation of the sub clock XT requires the connection of an external resonator to XT1 and XT2.
This clock is mainly used for real-time clock applications.

#### High Speed Internal Oscillator (HS IntOSC)

The HS IntOSC generates the clock RH, which runs at a frequency of 8 MHz (Typ.).

#### Low Speed Internal Oscillator (LS IntOSC)

The LS IntOSC generates the clock RL, which runs at a frequency of 240 kHz (Typ.). It starts operation at power up and cannot be stopped, hence it is always operating.

#### PLL

The PLL circuits generate high speed operation clocks CPLLOUT and PPLLOUT for normal operation of the microcontroller.

The clocks supplied by the clock oscillators (X, XT, RH, RL, CPLLOUT, PPLLOUT) and their divided clocks (CPUCLK, CPUCLK2, CPUCLK4, PPLLCLK, PPLLCLK2, and PPLLCLK4) are all generated in the clock generation circuit.

### 12.2.2 Clock Selection

The clocks generated by the clock oscillators are input to the clock selectors CKSC_A<name>/ CKSC_I<name>.

Domain clocks CKSCLK_A<name>/CKSCLK_I<name> are selected by dedicated clock selectors from clocks directly input from the oscillators, or in some cases from clocks that have been divided by clock dividers.

- CKSC_I<name>S_CTL/CKSC_I<name>D_CTL registers: determine the clock for the Isolated area clock domains.
- CKSC_A<name>S_CTL/CKSC_A<name>D_CTL registers: determine the clock for the Always-On area clock domains.

Note that not all available clocks generated by the clock oscillators are input to each clock selector.

The following clocks are supplied to the CPU and related modules from the clock generation circuit.

#### Emergency Clock (EMCLK)

The emergency clock EMCLK is supplied by the

- HS IntOSC, if it is active
- LS IntOSC, if the HS IntOSC is inactive

The selection is done automatically after CLMA0 reset is occurred, so if the HS IntOSC becomes lower than the limit for any reason, vital modules of the microcontroller are still in operation, since the LS IntOSC does not stop.

#### CPU Subsystem Clock (CPUCLK)

The CPU Subsystem clock CPUCLK is derived from PLL clock CPLLOUT, MainOSC, and EMCLK. The CPU clock selector CKSC_CPUCLK incorporates the selector CPUCLKS, followed by the clock divider CPUCLKD.

The CPUCLK clock divider provides the frequency-divided CPUCLK2 clock signal and CPUCLK4 clock signal derived from CPUCLK.

### 12.2.3 Clock Domains

The clock controller allows selection of the respective clocks for the CPU and peripheral modules. The clock control scope is called the clock domain. For the correspondence between the CPU and peripheral modules and clock domains, see **Section 12.5.3, Clock Domain Settings**.

### 12.2.4 Resetting Clock Oscillators

The clock oscillators on the Always-On area are reset by the PURES signal.

The HS IntOSC is reset when $\overline{\text{CLMA0RES}}$ is generated and the MainOSC is reset when $\overline{\text{CLMA1RES}}$ is generated.

The clock oscillator on the Isolated area is reset by the ISORES signal.

For further details on the clock oscillators, see **Section 12.3, Clock Oscillators**.

**CAUTION**

**For the specifications of the frequencies, acceptable variation, and other parameters of the clock generators, see Section 40, Electrical Characteristics.**

# 12.3 Clock Oscillators

## 12.3.1 Main Oscillator (MainOSC)

The Main Oscillator generates the clock X. X is also used as the PLL input clock PLLCLKIN.

**Figure 12.3** shows the basic configuration and signals of the MainOSC.

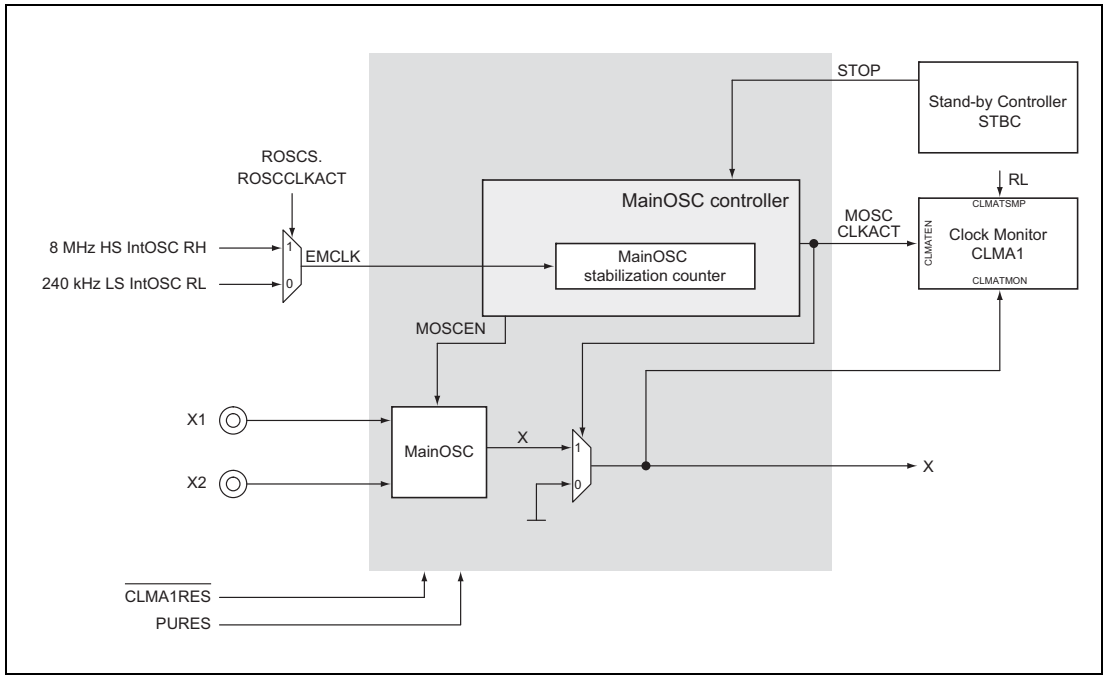



**Figure 12.3 Main Oscillator (MainOSC)**

### MainOSC

The MainOSC stops operating after reset is released. To use the MainOSC, set the MainOSC enable trigger bit (MOSCE.MOSCENTRG) to 1 to start the MainOSC.

### MainOSC stabilization

The MOSCST.MOSCCLKST[16:0] bits set the MainOSC oscillation stabilization time.
The MainOSC stabilization counter counts the oscillation stabilization time with EMCLK as the clock source for counting. The oscillation stabilization time can be set to up to $2^{17}$ –1 EMCLK cycles.
As long as the MainOSC is not stable, the MOSCCLKACT signal disables the X output.
When the MainOSC stabilization counter reaches the value specified in MOSCST.MOSCCLKST[16:0], X is assumed to be stable and MOSCCLKACT switches from 0 to 1 to enable output of X when a waveform is output from MainOSC.

Stable and active X clock is indicated by MOSCS.MOSCCLKACT = 1.

### MainOSC amplification gain

By using MOSCC.MOSCAMPSEL[1:0], the MainOSC's input frequency, determined by the external resonator, can be selected from 16MHz, 20 MHz, and 24 MHz.

### MainOSC STOP requests in stand-by mode

The STOP signal from the Stand-by Controller requests the MainOSC Controller to switch off the X clock in stand-by modes (STOP mode, DeepSTOP mode and Cyclic STOP mode).

The stop request mask bit MOSCSTPM.MOSCSTPMSK controls whether the MainOSC is stopped during stand-by or continues operation:

- MOSCSTPM.MOSCSTPMSK = 0:
  The STOP request signal is not masked, so the MainOSC is stopped in stand-by.
  If the MainOSC is in operation before stand-by, it is automatically re-started after wake-up from stand-by, and the MainOSC stabilization counter counts the oscillation stabilization time.
  However, the STOP request is masked under the following conditions, even if MOSCSTPM.MOSCSTPMSK = 0. Therefore, the MainOSC will continue to operate even in stand-by mode.
  - If the stop mask is set (CKSC_xxxx_STPM = 0000 $0003_H$) for a clock domain for which the MainOSC is selected.
- MOSCSTPM.MOSCSTPMSK = 1:
  The STOP request signal is not masked, so the MainOSC continues to operate in stand-by.

### Clock monitor control

The MainOSC activity signal MOSCCLKACT enables or disables supervision by the Clock Monitor CLMA1. In case the MainOSC is inactive (MOSCCLKACT = 0), supervision of its output clock X by CLMA1 is also disabled.

### MainOSC enable/disable trigger

The MainOSC can be enabled and disabled by the enable and disable trigger control bits:

- Enable trigger MOSCE.MOSCENTRG = 1 starts the MainOSC.
  Note that setting the enable trigger is only effective if the MainOSC is inactive, i.e. if MOSCS.MOSCCLKACT = 0.
- Disable trigger MOSCE.MOSCDISTRG = 1 stops the MainOSC.
  Note that setting the disable trigger is only effective if the MainOSC is active (MOSCCLKACT = 1) and the MainOSC stop requests are not masked (MOSCSTPM.MOSCSTPMSK = 0).

### Direct clock input to X1

A clock waveform from an external clock source can be supplied to X1 pin. In this case, set the MOSCM bit of MOSCM register to 1 before clock input to X1 pin is supplied.

### 12.3.2 Sub Oscillator (SubOSC)

The Sub Oscillator generates the sub clock XT. XT has usually a frequency of 32.768 kHz and is used for the Real-time Clock.

**Figure 12.4** shows the basic structure and signals of the SubOSC.

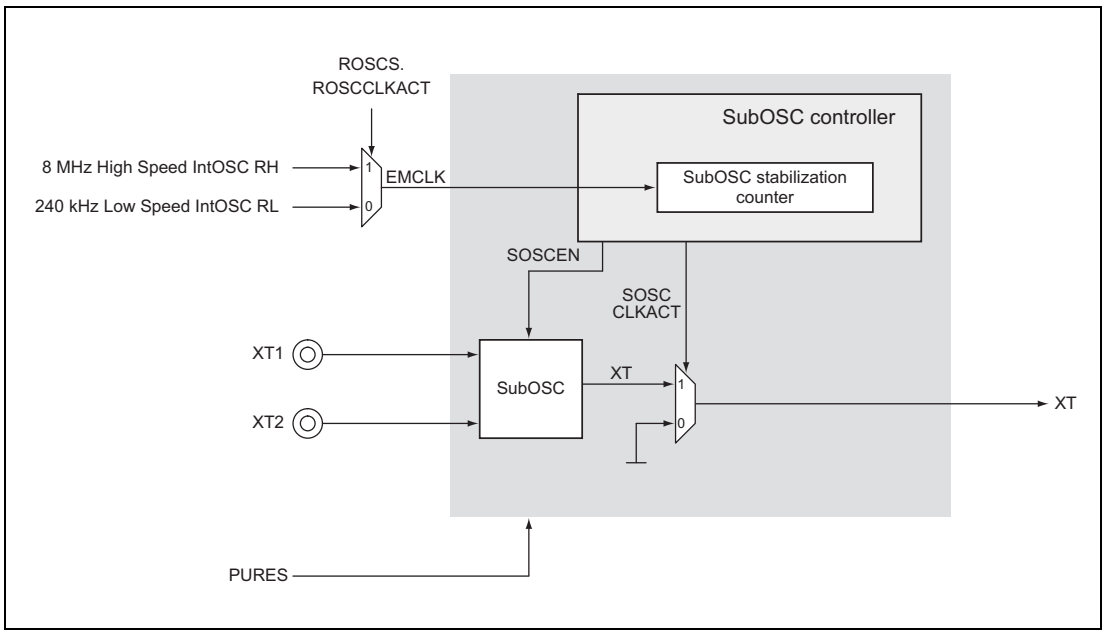



**Figure 12.4 Sub Oscillator (SubOSC)**

#### SubOSC enable

The SubOSC stops operating after reset is released. To use the SubOSC, set SubOSC enable trigger bit (SOSCE.SOSCENTRG) to 1 to start the SubOSC.

#### SubOSC stabilization

The SOSCST.SOSCCLKST[29:0] bits set the SubOSC oscillation stabilization time.
The SubOSC stabilization counter counts the oscillation stabilization time with EMCLK as the clock source for counting.
As long as the SubOSC is not stable, the SOSCCLKACT signal disables the XT output.
When the SubOSC stabilization counter reaches the value specified in SOSCST.SOSCCLKST[29:0], XT is assumed to be stable and SOSCCLKACT switches from 0 to 1 to enable output of XT.
Secure the stabilization time longer than 2 seconds.
Stable and active XT clock is indicated by SOSCS.SOSCCLKACT = 1.

#### SubOSC input frequencies

The SubOSC input frequency is 32.768 kHz (Typ.).

#### SubOSC enable trigger/disable trigger

SubOSC can be enabled or disabled by using enable/disable trigger control bit.

- Enable trigger SOSCE.SOSCENTRG = 1 starts the SubOSC.
  Note that setting the enable trigger is only effective if the SubOSC is inactive, i.e. if SOSCS.SOSCCLKACT = 0.

- Disable trigger SOSCE.SOSCDISTRG = 1 stops the SubOSC. Note that setting the disable trigger is only effective if the SubOSC is active; that is, if SOSCS.SOSCCLKACT = 1.

### 12.3.3 High Speed Internal Oscillator (HS IntOSC)

The High Speed Internal Oscillator generates the clock RH. RH has a nominal frequency of 8 MHz.

**Figure 12.5** shows the basic configuration and signals of the HS IntOSC.

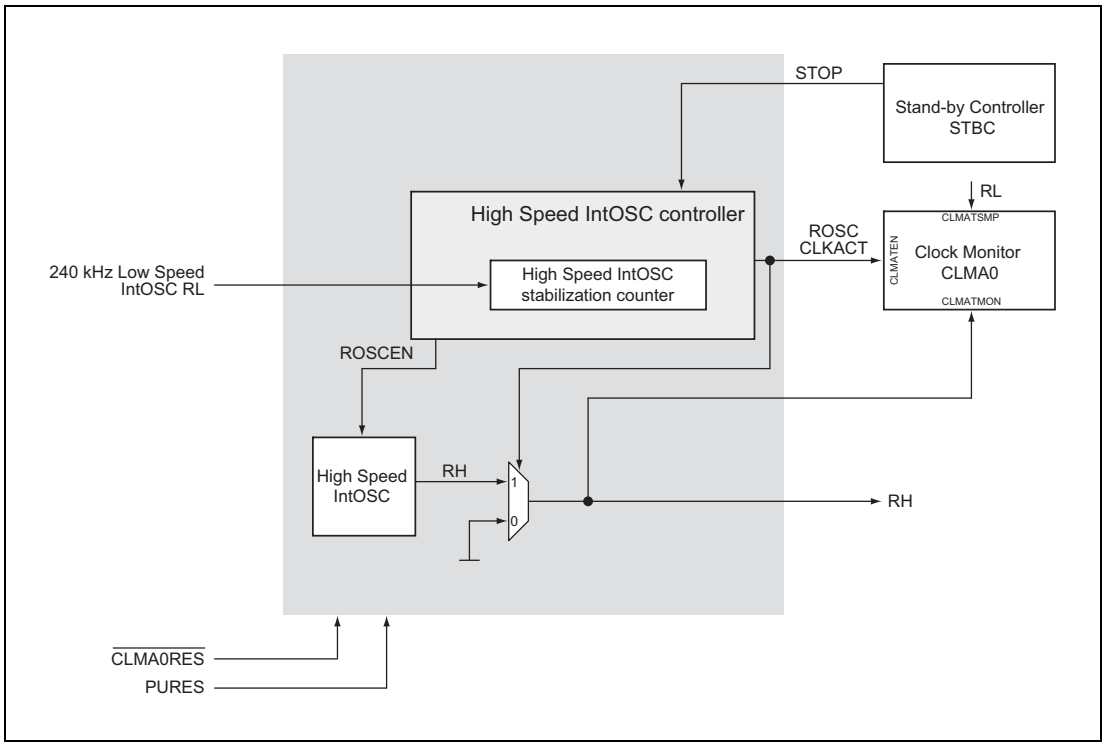



**Figure 12.5 High Speed Internal Oscillator (HS IntOSC)**

After reset release the HS IntOSC starts operation.

**NOTE**

The HS IntOSC can neither be stopped nor started by software. It can only be stopped in stand-by mode. On the other hand, when CLMA0 is reset, the HS IntOSC can be enabled to stop by software.

#### HS IntOSC stabilization

HS IntOSC outputs RH when it is stabilized.
Stable and active RH clock is indicated by ROSCS.ROSCCLKACT = 1.

#### HS IntOSC STOP requests in stand-by mode

The STOP signal from the Stand-By Controller requests the HS IntOSC Controller to switch off the RH clock in stand-by modes (STOP mode, DeepSTOP mode and Cyclic STOP mode).
The stop request mask bit ROSCSTPM.ROSCSTPMSK controls whether the HS IntOSC is stopped during stand-by or continues operation:

- ROSCSTPM.ROSCSTPMSK = 0:
  The STOP request signal is not masked, so the HS IntOSC is stopped during stand-by and automatically restarted after wake-up from stand-by.
  However, the STOP request is masked under the following conditions, even if ROSCSTPM.ROSCSTPMSK = 0. Therefore, the HS IntOSC will continue to operate even in stand-by mode.
  - If the stop mask is set (CKSC_xxxx_STPM = $0000_0003_H$) for a clock domain for which the HS IntOSC is selected.
  - If the low power sampler (LPS) is operating
- ROSCSTPM.ROSCSTPMSK = 1:
  The STOP request signal is masked, so the HS IntOSC continues to operate during stand-by.

### Clock Monitor control

The HS IntOSC activity signal ROSCCLKACT enables or disables supervision by the Clock Monitor CLMA0. In case the HS IntOSC is inactive (ROSCCLKACT = 0), supervision of its output clock by CLMA0 is also deactivated.

The HS IntOSC clock RH is used as the sampling clock for Clock Monitor CLMA2.

### HS IntOSC disable trigger

The disable trigger, ROSCE.ROSCDISTRG = 1 stops the HS IntOSC.

The setting of the disable trigger is enabled when HS IntOSC is active (ROSCS.ROSCCLKACT = 1) and HS IntOSC stop requests are not masked (ROSCSTPM.ROSCSTPMSK = 0).

### HS IntOSC user calibration function

The HS IntOSC user trimming register (ROSCUT) enables adjustment of the HS IntOSC frequency. The initial value of ROSCUT is the preset value of the HS IntOSC trimming data. Overwrite the value with “read value + 1” or “read value – 1” until the HS IntOSC frequency reaches the target frequency range.

### 12.3.4 Low Speed Internal Oscillator (LS IntOSC)

The Low Speed Internal Oscillator generates the clock RL. RL has a nominal frequency of 240 kHz.

**Figure 12.6** shows the basic configuration and signals of the LS IntOSC.

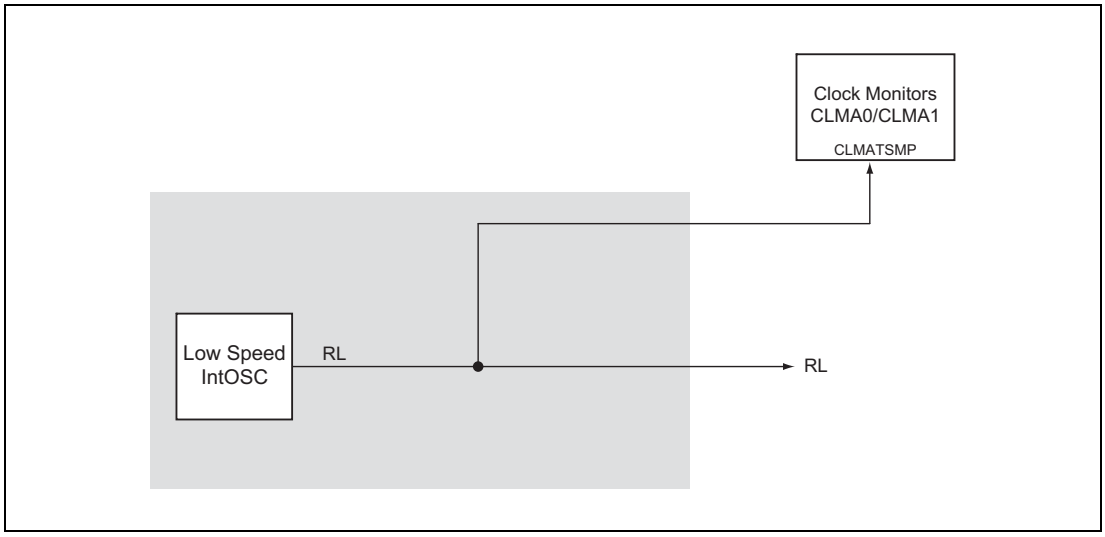



**Figure 12.6 Low Speed Internal Oscillator (LS IntOSC)**

After reset release the LS IntOSC starts operation. It cannot be stopped.

The LS IntOSC clock RL is used as the sampling clock for the Clock Monitors CLMA0 and CLMA1.

### 12.3.5 PLL

The MainOSC or HS IntOSC is input to the Phase-Locked Loops (PLL) clock oscillator as PLLCLKIN. The PLL output clocks CPLLOUT and PPLLOUT serve as the main operation clocks for the microcontroller.

**Figure 12.7** shows the basic configuration and signals of the PLL.

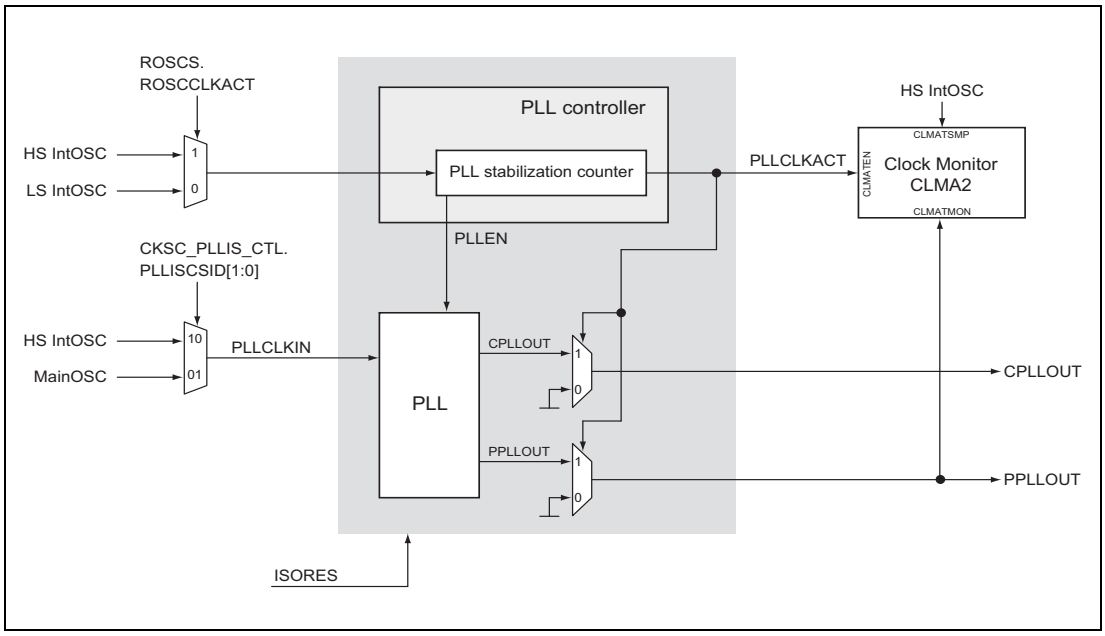



Figure 12.7 PLL

#### PLL enable

The PLL stops operating after reset is released. To use the PLL, set the PLL enable trigger bit (PLLE.PLLENTRG) to 1 to start the PLL.

#### PLL stabilization

The PLL stabilization counter starts counting the stabilization time.
As long as the PLL is not stable, the PLLCLKACT signal disables the PPLLOUT and CPLLOUT outputs.
When the PLL stabilization counter reaches the predefined value, PPLLOUT and CPLLOUT are assumed to be stable and PLLCLKACT switches from 0 to 1 to enable output of PPLLOUT and CPLLOUT.

The stable and active state of the PPLLOUT and CPLLOUT clocks is indicated by PLLS.PLLCLKACT = 1.

#### PLL in stand-by modes

In STOP mode, the PLL is automatically disabled and resumes operation after wake-up from STOP mode, if it was operating before entering STOP mode.

The PLL is also automatically disabled when transitioning to DeepSTOP mode. However, after restoring from DeepSTOP mode, the PLL needs to be reconfigured.

In Cyclic RUN and Cyclic STOP mode, the PLL is not available. Do not enable the PLL by using the PLL Enable register in Cyclic RUN mode.

### Clock Monitor control

The PLL activity signal PLLCLKACT enables or disables supervision by the Clock Monitor CLMA2. In case the PLL is inactive (PLLCLKACT = 0), supervision of the output clock PPLLOUT by CLMA2 is also deactivated.

### PLL enable/disable trigger

The PLL can be enabled and disabled by the enable and disable trigger control bits:

- Enable trigger PLLE.PLLENTRG = 1 starts the PLL
  Note that setting the enable trigger is only effective if the PLL is inactive, i.e. if PLLS.PLLCLKACT = 0.
- Disable trigger PLLE.PLLDISTRG = 1 stops the PLL
  Note that setting the disable trigger is only effective if the PLL is active, i.e. if PLLS.PLLCLKACT = 1.

### PLL input clock (PLLCLKIN) selection

The PLL input clock (PLLCLKIN) can be selected from MainOSC and HS IntOSC by using the CKSC_PLLIS_CTL register.

The maximum frequency of CPLLOUT and PPLLOUT is limited when the HS IntOSC is selected as PLL input clock.

### 12.3.5.1 PLL Parameters

The PLL is configured by a set of parameters, loaded from the control register PLLC and CKSC_CPUCLKD_CTL.

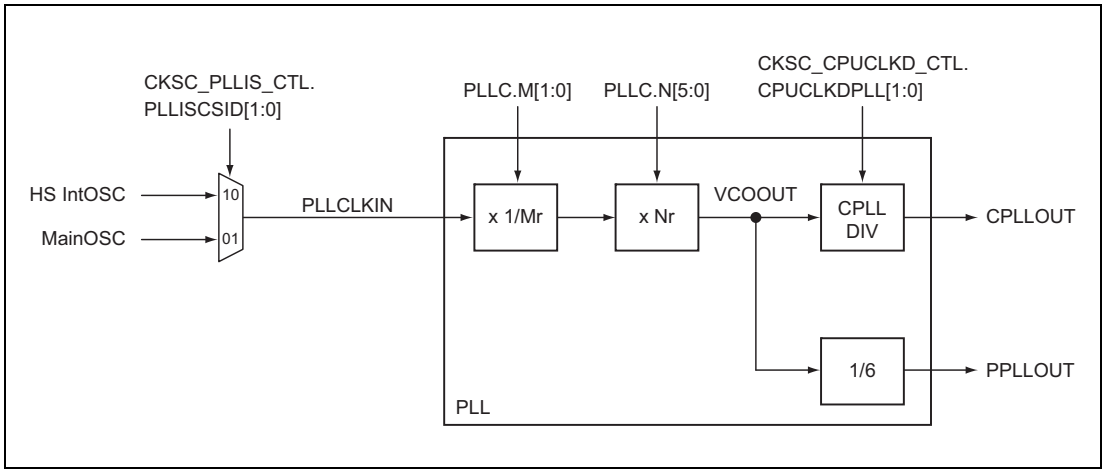



**Figure 12.8 PLL Circuit**

#### CPLLOUT and PPLLOUT

The PLL has two clock outputs: “CPLLOUT” and “PPLLOUT”. CPLLOUT is one of the clock sources of the CPU subsystem, and PPLLOUT is one of the clock sources of the peripheral modules. CPLLOUT and PPLLOUT share the same clock source “VCOOUT”, which is the output of the voltage controlled oscillator (VCO). The clock frequency of VCOOUT is calculated by the following formula:

$$f_{VCOOUT} = f_{PLLCLKIN} \times (Nr / Mr)$$

The clock frequency of CPLLOUT “$f_{CPLL}$” and that of PPLLOUT “$f_{PPLL}$” are integer fractions of the VCO output frequency $f_{VCOOUT}$. $f_{CPLL}$ and $f_{PPLL}$ are calculated by the following formulas:

$$f_{CPLL} = f_{VCOOUT} \times 1/Pr = f_{PLLCLKIN} \times (Nr / Mr) \times 1/Pr$$

$$f_{PPLL} = f_{VCOOUT} \times 1/6 = f_{PLLCLKIN} \times (Nr / Mr) \times 1/6$$

The values Nr and Mr are derived from PLLC register bits:

$$Nr = PLLC.N[5:0] + 1$$

$$Mr = PLLC.M[1:0] + 1$$

The setting range of Mr: $1 \leq Mr \leq 3$

The value Pr is derived from CKSC_CPUCLKD_CTL. CPUCLKDPLL[1:0], and the value is 4, 5, or 6.

# 12.4 Registers

## 12.4.1 List of Registers

The registers of the clock controller are listed below.

**Table 12.1 List of Clock Controller Registers (1/2)**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| CLKCTL | **Clock oscillator registers:** | | |
| | MainOSC enable register | MOSCE | FFF8 1100$_H$ |
| | MainOSC status register | MOSCS | FFF8 1104$_H$ |
| | MainOSC control register | MOSCC | FFF8 1108$_H$ |
| | MainOSC stabilization time register | MOSCST | FFF8 110C$_H$ |
| | MainOSC stop mask register | MOSCSTPM | FFF8 1118$_H$ |
| | MOSC mode control register | MOSCM | FFF8 111C$_H$ |
| | SubOSC enable register | SOSCE*1 | FFF8 1200$_H$ |
| | SubOSC status register | SOSCS*1 | FFF8 1204$_H$ |
| | SubOSC stabilization time register | SOSCST*1 | FFF8 120C$_H$ |
| | HS IntOSC enable register | ROSCE | FFF8 1000$_H$ |
| | HS IntOSC status register | ROSCS | FFF8 1004$_H$ |
| | HS IntOSC stop mask register | ROSCSTPM | FFF8 1018$_H$ |
| | PLL enable register | PLLE | FFF8 9000$_H$ |
| | PLL status register | PLLS | FFF8 9004$_H$ |
| | PLL control register | PLLC | FFF8 9008$_H$ |
| | PLL input clock selection register | CKSC_PLLIS_CTL | FFF8 A700$_H$ |
| | PLL input clock active register | CKSC_PLLIS_ACT | FFF8 A708$_H$ |
| | PPLLCLK source clock selection register | CKSC_PPLLCLKS_CTL | FFF8 A010$_H$ |
| | PPLLCLK source clock active register | CKSC_PPLLCLKS_ACT | FFF8 A018$_H$ |
| | HS IntOSC user trimming register | ROSCUT | FFF8 101C$_H$ |
| | **Clock selectors registers:** | | |
| | C_AWO_WDTA clock divider register | CKSC_AWDTAD_CTL | FFF8 2000$_H$ |
| | C_AWO_WDTA clock divider active register | CKSC_AWDTAD_ACT | FFF8 2008$_H$ |
| | C_AWO_WDTA stop mask register | CKSC_AWDTAD_STPM | FFF8 2018$_H$ |
| | C_AWO_TAUJ source clock selection register | CKSC_ATAUJS_CTL | FFF8 2100$_H$ |
| | C_AWO_TAUJ source clock active register | CKSC_ATAUJS_ACT | FFF8 2108$_H$ |
| | C_AWO_TAUJ clock divider register | CKSC_ATAUJD_CTL | FFF8 2200$_H$ |
| | C_AWO_TAUJ clock divider active register | CKSC_ATAUJD_ACT | FFF8 2208$_H$ |
| | C_AWO_TAUJ stop mask register | CKSC_ATAUJD_STPM | FFF8 2218$_H$ |
| | C_AWO_RTCA source clock selection register | CKSC_ARTCAS_CTL*1 | FFF8 2300$_H$ |
| | C_AWO_RTCA source clock active register | CKSC_ARTCAS_ACT*1 | FFF8 2308$_H$ |
| | C_AWO_RTCA clock divider register | CKSC_ARTCAD_CTL*1 | FFF8 2400$_H$ |
| | C_AWO_RTCA clock divider active register | CKSC_ARTCAD_ACT*1 | FFF8 2408$_H$ |
| | C_AWO_RTCA stop mask register | CKSC_ARTCAD_STPM*1 | FFF8 2418$_H$ |
| | C_AWO_ADCA source clock selection register | CKSC_AADCAS_CTL | FFF8 2500$_H$ |
| | C_AWO_ADCA source clock active register | CKSC_AADCAS_ACT | FFF8 2508$_H$ |
| | C_AWO_ADCA clock divider register | CKSC_AADCAD_CTL | FFF8 2600$_H$ |
| | C_AWO_ADCA clock divider active register | CKSC_AADCAD_ACT | FFF8 2608$_H$ |

Table 12.1 List of Clock Controller Registers (2/2)

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| CLKCTL | C_AWO_ADCA stop mask register | CKSC_AADCAD_STPM | FFF8 $2618_H$ |
| | C_AWO_FOUT source clock selection register | CKSC_AFOUTS_CTL | FFF8 $2700_H$ |
| | C_AWO_FOUT source clock active register | CKSC_AFOUTS_ACT | FFF8 $2708_H$ |
| | C_AWO_FOUT stop mask register | CKSC_AFOUTS_STPM | FFF8 $2718_H$ |
| | C_ISO_CPUCLK source clock selection register | CKSC_CPUCLKS_CTL | FFF8 $A000_H$ |
| | C_ISO_CPUCLK source clock active register | CKSC_CPUCLKS_ACT | FFF8 $A008_H$ |
| | C_ISO_CPUCLK clock divider register | CKSC_CPUCLKD_CTL | FFF8 $A100_H$ |
| | C_ISO_CPUCLK clock divider active register | CKSC_CPUCLKD_ACT | FFF8 $A108_H$ |
| | C_ISO_PERI1 source clock selection register | CKSC_IPERI1S_CTL | FFF8 $A200_H$ |
| | C_ISO_PERI1 source clock active register | CKSC_IPERI1S_ACT | FFF8 $A208_H$ |
| | C_ISO_PERI2 source clock selection register | CKSC_IPERI2S_CTL | FFF8 $A300_H$ |
| | C_ISO_PERI2 source clock active register | CKSC_IPERI2S_ACT | FFF8 $A308_H$ |
| | C_ISO_LIN source clock selection register | CKSC_ILINS_CTL | FFF8 $A400_H$ |
| | C_ISO_LIN source clock active register | CKSC_ILINS_ACT | FFF8 $A408_H$ |
| | C_ISO_ADCA source clock selection register | CKSC_IADCAS_CTL*1 | FFF8 $A500_H$ |
| | C_ISO_ADCA source clock active register | CKSC_IADCAS_ACT*1 | FFF8 $A508_H$ |
| | C_ISO_ADCA clock divider register | CKSC_IADCAD_CTL*1 | FFF8 $A600_H$ |
| | C_ISO_ADCA clock divider active register | CKSC_IADCAD_ACT*1 | FFF8 $A608_H$ |
| | C_ISO_LIN clock divider register | CKSC_ILIND_CTL | FFF8 $A800_H$ |
| | C_ISO_LIN clock divider active register | CKSC_ILIND_ACT | FFF8 $A808_H$ |
| | C_ISO_LIN stop mask register | CKSC_ILIND_STPM | FFF8 $A818_H$ |
| | C_ISO_CAN source clock selection register | CKSC_ICANS_CTL | FFF8 $A900_H$ |
| | C_ISO_CAN source clock active register | CKSC_ICANS_ACT | FFF8 $A908_H$ |
| | C_ISO_CAN stop mask register | CKSC_ICANS_STPM | FFF8 $A918_H$ |
| | C_ISO_CANOSC clock divider register | CKSC_ICANOSCD_CTL | FFF8 $AA00_H$ |
| | C_ISO_CANOSC clock divider active register | CKSC_ICANOSCD_ACT | FFF8 $AA08_H$ |
| | C_ISO_CANOSC stop mask register | CKSC_ICANOSCD_STPM | FFF8 $AA18_H$ |
| | C_ISO_CSI source clock selection register | CKSC_ICSIS_CTL | FFF8 $AB00_H$ |
| | C_ISO_CSI source clock active register | CKSC_ICSIS_ACT | FFF8 $AB08_H$ |
| | C_ISO_IIC source clock selection register | CKSC_IIICS_CTL | FFF8 $AC00_H$ |
| | C_ISO_IIC source clock active register | CKSC_IIICS_ACT | FFF8 $AC08_H$ |

Note 1. Supported only by 144 pin and 176 pin products.

## 12.4.2 Clock Oscillator Registers

### 12.4.2.1 MOSCE — MainOSC Enable Register

This register is used to start and stop the MainOSC.

The correct write sequence using the PROTCMD0 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by the power-up reset signal PURES and $\overline{\text{CLMA1RES}}$.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 1100$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MOSCDISTRG | MOSCENTRG |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.2 MOSCE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | MOSCDISTRG | MainOSC Disable Trigger*1,*3<br>MOSCSTPM.MOSCSTPMSK = 0<br>0: No function<br>1: Stops MainOSC<br><br>MOSCSTPM.MOSCSTPMSK=1<br>When writing this bit while MOSCSTPM.MOSCSTPMSK = 1, the write value should be 0.<br>0: No function<br>1: Setting Prohibited<br>This bit is automatically cleared to 0 by hardware after MainOSC is disabled. |
| 0 | MOSCENTRG | MainOSC Enable Trigger*2,*3<br>0: No function<br>1: Starts MainOSC<br>This bit is automatically cleared to 0 by hardware after MainOSC is enabled. |

Note 1. Follow the procedure given below for stopping the MainOSC by using MOSCDISTRG.

1. Confirm that the MainOSC is neither waiting for being enabled nor stopped (both MOSCE.MOSCDISTRG = 0 and MOSCE.MOSCENTRG = 0).
2. Confirm that the MainOSC is active (MOSCS.MOSCCLKACT = 1).
3. Check that there is no clock domain for which the MainOSC is selected. If the MainOSC is selected for a clock domain, disable the setting or select a clock source other than MainOSC.
4. Confirm that the MainOSC stop mask register is NOT set to "MainOSC continues operation in stand-by mode". Otherwise, set the register to “MainOSC stops operation in stand-by mode” (MOSCSTPM.MOSCSTPMSK = 0).
5. Stop the MainOSC (MOSCE.MOSCDISTRG = 1).
6. Confirm that the MainOSC has been stopped (MOSCS.MOSCCLKACT = 0).

Note 2. Follow the procedure given below for starting the MainOSC by using MOSCENTRG

1. Confirm that the MainOSC is neither waiting for being enabled nor stopped

(both MOSCE.MOSCDISTRG = 0 and MOSCE.MOSCENTRG = 0).

2. Confirm that the MainOSC is inactive (MOSCS.MOSCCLKACT = 0).
3. Start the MainOSC (MOSCE.MOSCENTRG = 1).
4. Confirm that the MainOSC has been started (MOSCS.MOSCCLKACT = 1).

Note 3. Starting and stopping the MainOSC at the same time by using the start and stop bits, i.e., by setting MOSCE.MOSCENTRG = 1 and MOSCE.MOSCDISTRG = 1 at the same time is not allowed.

### 12.4.2.2 MOSCS — MainOSC Status Register

This register provides active status information about the MainOSC.

This register is initialized by the power-up reset signal PURES and $\overline{\text{CLMA1RES}}$.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 $1104_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1*1 | 0*1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | MOSCCLKACT | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. The values of bit 1 and 0 are undefined.
After masking bit 1 and 0, check only bit 2 to verify the status.

**Table 12.3 MOSCS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 | MOSCCLKACT | MainOSC Active Status<br>0: MainOSC is inactive<br>1: MainOSC is active |
| 1, 0 | Reserved | When read, an undefined value is returned. |

### 12.4.2.3 MOSCC — MainOSC Control Register

This register is used to specify amplification gain of the MainOSC.

This register is initialized by the power-up reset signal PURES and $\overline{\text{CLMA1RES}}$.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 $1108_H$

**Value after reset:** 0000 $0004_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MOSCAMPSEL [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.4 MOSCC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | MOSC AMPSEL[1:0] | MainOSC Amplification Gain Selection<br>00: Amplification gain for fx = 24MHz<br>01: Amplification gain for fx = 20MHz<br>10: Amplification gain for fx = 16MHz<br>11: Setting prohibited |

**CAUTION**

Set this register when MainOSC is stopped.

### 12.4.2.4 MOSCST — MainOSC Stabilization Time Register

This register determines the MainOSC stabilization time.

This register is initialized by the power-up reset signal PURES and $\overline{\text{CLMA1RES}}$.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 110C$_H$

**Value after reset:** 0000 44C0$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MOSCCLKST16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MOSCCLKST[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 12.5 MOSCST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 17 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 16 to 0 | MOSC CLKST[16:0] | The MOSCCLKST[16:0] bits specify the count value for the MainOSC stabilization counter.<br>• If HS IntOSC active (ROSCS.ROSCCLKACT = 1): Stabilization time = MOSCCLKST[16:0] / $f_{RH}$<br>• If HS IntOSC inactive (ROSCS.ROSCCLKACT = 0): Stabilization time = MOSCCLKST[16:0] / $f_{RL}$ |

**NOTE**

See **Section 40, Electrical Characteristics** for information about the MainOSC stabilization time.

**CAUTION**

Set this register when MainOSC is stopped.

### 12.4.2.5 MOSCSTPM — MainOSC Stop Mask Register

This register is initialized by the power-up reset signal PURES and $\overline{\text{CLMA1RES}}$.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 1118$_H$

**Value after reset:** 0000 0002$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MOSCSTPMSK |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 12.6 MOSCSTPM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | MOSCSTPMSK | MainOSC Stop Request Mask<br>0: MainOSC stops operation in stand-by mode.<br>MainOSC stops operation in the case the MainOSC disable trigger MOSCE.MOSCDISTRG is set to 1.<br>1: MainOSC continues operation in stand-by mode.<br>MainOSC continues operation even in the case the MainOSC disable trigger MOSCE.MOSCDISTRG is set to 1. |

### 12.4.2.6 MOSCM — MainOSC Mode Control Register

This register is initialized by the power-up reset signal PURES and $\overline{\text{CLMA1RES}}$.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 111C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MOSCM |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 12.7 MOSCM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | MOSCM | MainOSC Mode control<br>0: OSC mode. (default)<br>1: EXCLK mode. MainOSC amplifier is disabled. |

**CAUTION**

Set this register when MainOSC is stopped.

### 12.4.2.7 SOSCE — SubOSC Enable Register

This register is used to start and stop the SubOSC.

The correct write sequence using the PROTCMD0 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by the power-up reset signal PURES.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 1200$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SOSCDISTRG | SOSCENTRG |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.8 SOSCE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | SOSCDISTRG | SubOSC Disable Trigger*1,*2,*3<br>0: No function<br>1: Stops SubOSC<br>This bit is automatically cleared to 0 by hardware after SubOSC is disabled. |
| 0 | SOSCENTRG | SubOSC Enable Trigger*3<br>0: No function<br>1: Starts SubOSC<br>This bit is automatically cleared to 0 by hardware after SubOSC is enabled. |

Note 1. Follow the procedure given below for stopping the SubOSC by using SOSCDISTRG.

1. Confirm that the SubOSC is neither waiting for being enabled nor stopped (SOSCE.SOSCDISTRG = 0 and SOSCE.SOSCENTRG = 0).
2. Confirm that the SubOSC is active (SOSCS.OSCCLKACT = 1).
3. Check that there is no clock domain for which the SubOSC is selected. If the SubOSC is selected for a clock domain, disable the setting or select a clock source other than the SubOSC.
4. Stop the SubOSC (SOSCE.SOSCDISTRG = 1).
5. Confirm that the SubOSC has been stopped (SOSCS.SOSCCLKACT = 0).

Note 2. Follow the procedure given below for starting the SubOSC by using SOSCDISTRG.

1. Confirm that the SubOSC is neither waiting for being enabled nor stopped (SOSCE.SOSCDISTRG = 0 and SOSCE.SOSCENTRG = 0).
2. Confirm that the SubOSC is inactive (SOSCS.OSCCLKACT = 0).
3. Start the SubOSC (SOSCE.SOSCENTRG = 1).
4. Confirm that the SubOSC has been started (SOSCS.SOSCCLKACT = 1).

Note 3. Starting and stopping the SubOSC at the same time by using the start and stop bits, i.e., by setting SOSCE.SOSCENTRG = 1 and SOSSCE.SOSCDISTRG = 1 at the same time is not allowed.

### 12.4.2.8 SOSCS — SubOSC Status Register

This register provides active status information about the SubOSC.

This register is initialized by the power-up reset signal PURES.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 1204$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1*1 | 0*1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | SOSCCLKACT | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. The values of bit 1 and 0 are undefined.
After masking bit 1 and 0, check only bit 2 to verify the status.

**Table 12.9 SOSCS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 | SOSCCLKACT | SubOSC Activation Status<br>0: SubOSC is inactive<br>1: SubOSC is active |
| 1, 0 | Reserved | When read, an undefined value is returned. |

### 12.4.2.9 SOSCST — SubOSC Stabilization Time Register

This register determines the SubOSC stabilization time.

This register is initialized by the power-up reset signal PURES.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 120C$_H$

**Value after reset:** 010C 8E00$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | SOSCCLKST[29:16] | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SOSCCLKST[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 12.10 SOSCST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 29 to 0 | SOSCCLKST [29:0] | The SOSCCLKST[29:0] bits specify the count value for the SubOSC stabilization time counter.<br>• If the HS IntOSC is active (ROSCS.ROSCCLKACT = 1): Stabilization time = SOSCCLKST[29:0] / $f_{RH}$<br>• If the HS IntOSC is inactive (ROSCS.ROSCCLKACT = 0): Stabilization time = SOSCCLKST[29:0] / $f_{RL}$ |

**NOTE**

See **Section 40, Electrical Characteristics** for information about the SubOSC stabilization time.

**CAUTION**

Set this register when SubOSC is stopped.

### 12.4.2.10 ROSCE — HS IntOSC Enable Register

This register is used to stop the HS IntOSC operation.

The correct write sequence using thePROTCMD0 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by the power-up reset signal PURES and $\overline{\text{CLMA0RES}}$.

**CAUTION**

Set the ROSCE.ROSCDISTRG bit only when the $\overline{\text{CLMA0RES}}$ has occurred. In other cases, setting this bit is prohibited.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 $1000_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ROSCDISTRG | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R |

**Table 12.11 ROSCE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | ROSCDISTRG | HS IntOSC Disable Trigger<br>ROSCSTPM.ROSCSTPMSK = 0<br>0: No function<br>1: Stops HS IntOSC<br><br>ROSCSTPM.ROSCSTPMSK = 1<br>Setting prohibited |
| 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

### 12.4.2.11 ROSCS — HS IntOSC Status Register

This register provides active status information about the HS IntOSC.

This register is initialized by the power-up reset signal PURES and $\overline{\text{CLMA0RES}}$.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 $1004_H$

**Value after reset:** $0000\ 0007_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1*1 | 0*1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | ROSCCLKACT | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. The values of bit 1 and 0 are undefined.
After masking bit 1 and 0, check only bit 2 to verify the status.

**Table 12.12 ROSCS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 | ROSCCLKACT | HS IntOSC Active Status<br>0: HS IntOSC is inactive<br>1: HS IntOSC is active |
| 1, 0 | Reserved | When read, an undefined value is returned. |

### 12.4.2.12 ROSCSTPM — HS IntOSC Stop Mask Register

This register is initialized by the power-up reset signal PURES and $\overline{\text{CLMA0RES}}$.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 $1018_H$

**Value after reset:** 0000 $0002_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ROSCSTPMSK |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 12.13 ROSCSTPM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | ROSCSTPMSK | HS IntOSC Stop Request Mask<br>0: HS IntOSC stops operation in stand-by mode<br>1: HS IntOSC continues operation in stand-by mode<br>Do not set the HS IntOSC disable trigger ROSCE.ROSCDISTRG to 1 while ROSCSTPMSK bit is set to 1. |

### 12.4.2.13 PLLE — PLL Enable Register

The correct write sequence using the PROTCMD1 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 $9000_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PLLDIS TRG | PLLEN TRG |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.14 PLLE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | PLLDISTRG | PLL Disable Trigger*1,*4<br>0: No function<br>1: Stops PLL<br>This bit is automatically cleared to 0 by hardware after PLL is disabled. |
| 0 | PLLENTRG | PLL Enable Trigger*2,*3,*4<br>0: No function<br>1: Starts PLL<br>This bit is automatically cleared to 0 by hardware after PLL is activated. |

Note 1. Follow the procedure given below for stopping the PLL by using PLLDISTRG.

1. Confirm that the PLL is neither waiting for being enabled nor stopped (both PLLE.PLLDISTRG = 0 and PLLE.PLLENTRG = 0).
2. Confirm that the PLL is active (PLLS.PLLCLKACT = 1).
3. Check that there is no clock domain for which the PLL is selected. If the PLL is selected for a clock domain, disable the setting or select a clock source other than the PLL.
4. Stop the PLL (PLLE.PLLDISTRG = 1).
5. Confirm that the PLL has been stopped (PLLS.PLLCLKACT = 0).

Note 2. Before starting PLL using PLLENTRG, confirm that the PLL input clock (MainOSC or HS IntOSC, selected by the CKSC_PLLIS_CTL) is operating.

Note 3. Follow the procedure given below for starting the PLL by using PLLENTRG.

1. Confirm that the PLL is neither waiting for being enabled nor stopped (both PLLE.PLLDISTRG = 0 and PLLE.PLLENTRG = 0).
2. Confirm that the PLL is inactive (PLLS.PLLCLKACT = 0).
3. Start the PLL (PLLE.PLLENTRG = 1).
4. Confirm that the PLL has been started (PLLS.PLLCLKACT = 1).

Note 4. Starting and stopping the PLL at the same time by using the start and stop bits, i.e., by setting PLLE.PLLENTRG = 1 and PLLE.PLLDISTRG = 1 at the same time is not allowed.

### 12.4.2.14 PLLS — PLL Status Register

This register provides active status information about the PLL.

This register is initialized by all reset sources (ISORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 9004$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1*1 | 0*1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | PLLCLKACT | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. The values of bit 1 and 0 are undefined.
After masking bit 1 and 0, check only bit 2 to verify the status.

**Table 12.15 PLLS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 | PLLCLKACT | PLL Active Status<br>0: PLL is inactive<br>1: PLL is active |
| 1, 0 | Reserved | When read, an undefined value is returned. |

### 12.4.2.15 PLLC — PLL Control Register

This register is used to set the PLL VCO output clock frequency $f_{VCOOUT}$, shown in **Section 12.3.5.1, PLL Parameters**.

This register can only be written, if the PLL is disabled.

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 $9008_H$

**Value after reset:** 0001 $133B_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | M[1:0] | | — | — | — | — | — | N[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| R/W | R | R | R | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 12.16 PLLC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12, 11 | M[1:0] | Division ratio Mr is set.<br>For M[1:0] settings, see **Table 12.17, PLL Output Table**. |
| 10 to 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 to 0 | N[5:0] | Division ratio Nr is set.<br>For N[5:0] settings, see **Table 12.17, PLL Output Table**. |

**CAUTION**

Set this register when PLL is stopped.

**Table 12.17 PLL Output Table (1/3)**

| PLLCLKIN frequency $f_{PLLCLKIN}$ (MHz) | PLLC. M[1:0] (Mr) | PLLC. N[5:0] (Nr) | VCOOUT frequency $f_{VCOOUT}$ (MHz) | CPLLOUT frequency $f_{CPLLOUT}$ (MHz)*1 | | | PPLLOUT frequency $f_{PPLLOUT}$ (MHz) |
|---|---|---|---|---|---|---|---|
| | | | | VCOOUT × 1/4 (PREMIUM, ADVANCED) | VCOOUT× 1/5 (PREMIUM, ADVANCED) | VCOOUT × 1/6 (PREMIUM, ADVANCED, ECO) | |
| 16 (MainOSC) | $01_B$ (Mr = 2) | $3B_H$ (Nr = 60) | 480.0 | 120.0 | 96.0 | 80.0 | 80.0 |
| | | $3A_H$ (Nr = 59) | 472.0 | 118.0 | 94.4 | 78.7 | 78.7 |
| | | $39_H$ (Nr = 58) | 464.0 | 116.0 | 92.8 | 77.3 | 77.3 |
| | | $38_H$ (Nr = 57) | 456.0 | 114.0 | 91.2 | 76.0 | 76.0 |
| | | $37_H$ (Nr = 56) | 448.0 | 112.0 | 89.6 | 74.7 | 74.7 |
| | | $36_H$ (Nr = 55) | 440.0 | 110.0 | 88.0 | 73.3 | 73.3 |
| | | $35_H$ (Nr = 54) | 432.0 | 108.0 | 86.4 | 72.0 | 72.0 |
| | | $34_H$ (Nr = 53) | 424.0 | 106.0 | 84.8 | 70.7 | 70.7 |
| | | $33_H$ (Nr = 52) | 416.0 | 104.0 | 83.2 | 69.3 | 69.3 |
| | | $32_H$ (Nr = 51) | 408.0 | 102.0 | 81.6 | 68.0 | 68.0 |
| | | $31_H$ (Nr = 50) | 400.0 | 100.0 | 80.0 | 66.7 | 66.7 |
| | | $30_H$ (Nr = 49) | 392.0 | 98.0 | 78.4 | 65.3 | 65.3 |
| | | $2F_H$ (Nr = 48) | 384.0 | 96.0 | 76.8 | 64.0 | 64.0 |
| 20 (MainOSC) | $01_B$ (Mr = 2) | $2F_H$ (Nr = 48) | 480.0 | 120.0 | 96.0 | 80.0 | 80.0 |
| | | $2E_H$ (Nr = 47) | 470.0 | 117.5 | 94.0 | 78.3 | 78.3 |
| | | $2D_H$ (Nr = 46) | 460.0 | 115.0 | 92.0 | 76.7 | 76.7 |
| | | $2C_H$ (Nr = 45) | 450.0 | 112.5 | 90.0 | 75.0 | 75.0 |
| | | $2B_H$ (Nr = 44) | 440.0 | 110.0 | 88.0 | 73.3 | 73.3 |
| | | $2A_H$ (Nr = 43) | 430.0 | 107.5 | 86.0 | 71.7 | 71.7 |
| | | $29_H$ (Nr = 42) | 420.0 | 105.0 | 84.0 | 70.0 | 70.0 |
| | | $28_H$ (Nr = 41) | 410.0 | 102.5 | 82.0 | 68.3 | 68.3 |
| | | $27_H$ (Nr = 40) | 400.0 | 100.0 | 80.0 | 66.7 | 66.7 |
| | | $26_H$ (Nr = 39) | 390.0 | 97.5 | 78.0 | 65.0 | 65.0 |
| | | $25_H$ (Nr = 38) | 380.0 | 95.0 | 76.0 | 63.3 | 63.3 |

**Table 12.17 PLL Output Table (2/3)**

| PLLCLKIN frequency $f_{PLLCLKIN}$ (MHz) | PLLC. M[1:0] (Mr) | PLLC. N[5:0] (Nr) | VCOOUT frequency $f_{VCOOUT}$ (MHz) | CPLLOUT frequency $f_{CPLLOUT}$ (MHz)*1 | | | PPLLOUT frequency $f_{PPLLOUT}$ (MHz) |
|---|---|---|---|---|---|---|---|
| | | | | VCOOUT × 1/4 (PREMIUM, ADVANCED) | VCOOUT× 1/5 (PREMIUM, ADVANCED) | VCOOUT × 1/6 (PREMIUM, ADVANCED, ECO) | |
| 24 (MainOSC) | $01_B$ (Mr = 2) | $27_H$ (Nr = 40) | 480.0 | 120.0 | 96.0 | 80.0 | 80.0 |
| | | $26_H$ (Nr = 39) | 468.0 | 117.0 | 93.6 | 78.0 | 78.0 |
| | | $25_H$ (Nr = 38) | 456.0 | 114.0 | 91.2 | 76.0 | 76.0 |
| | | $24_H$ (Nr = 37) | 444.0 | 111.0 | 88.8 | 74.0 | 74.0 |
| | | $23_H$ (Nr = 36) | 432.0 | 108.0 | 86.4 | 72.0 | 72.0 |
| | | $22_H$ (Nr = 35) | 420.0 | 105.0 | 84.0 | 70.0 | 70.0 |
| | | $21_H$ (Nr = 34) | 408.0 | 102.0 | 81.6 | 68.0 | 68.0 |
| | | $20_H$ (Nr = 33) | 396.0 | 99.0 | 79.2 | 66.0 | 66.0 |
| | | $1F_H$ (Nr = 32) | 384.0 | 96.0 | 76.8 | 64.0 | 64.0 |
| | $10_B$ (Mr = 3) | $3B_H$ (Nr = 60) | 480.0 | 120.0 | 96.0 | 80.0 | 80.0 |
| | | $3A_H$ (Nr = 59) | 472.0 | 118.0 | 94.4 | 78.7 | 78.7 |
| | | $39_H$ (Nr = 58) | 464.0 | 116.0 | 92.8 | 77.3 | 77.3 |
| | | $38_H$ (Nr = 57) | 456.0 | 114.0 | 91.2 | 76.0 | 76.0 |
| | | $37_H$ (Nr = 56) | 448.0 | 112.0 | 89.6 | 74.7 | 74.7 |
| | | $36_H$ (Nr = 55) | 440.0 | 110.0 | 88.0 | 73.3 | 73.3 |
| | | $35_H$ (Nr = 54) | 432.0 | 108.0 | 86.4 | 72.0 | 72.0 |
| | | $34_H$ (Nr = 53) | 424.0 | 106.0 | 84.8 | 70.7 | 70.7 |
| | | $33_H$ (Nr = 52) | 416.0 | 104.0 | 83.2 | 69.3 | 69.3 |
| | | $32_H$ (Nr = 51) | 408.0 | 102.0 | 81.6 | 68.0 | 68.0 |
| | | $31_H$ (Nr = 50) | 400.0 | 100.0 | 80.0 | 66.7 | 66.7 |
| | | $30_H$ (Nr = 49) | 392.0 | 98.0 | 78.4 | 65.3 | 65.3 |
| | | $2F_H$ (Nr = 48) | 384.0 | 96.0 | 76.8 | 64.0 | 64.0 |

**Table 12.17 PLL Output Table (3/3)**

| PLLCLKIN frequency $f_{PLLCLKIN}$ (MHz) | PLLC. M[1:0] (Mr) | PLLC. N[5:0] (Nr) | VCOOUT frequency $f_{VCOOUT}$ (MHz) | CPLLOUT frequency $f_{CPLLOUT}$ (MHz)*1 | | | PPLLOUT frequency $f_{PPLLOUT}$ (MHz) |
|---|---|---|---|---|---|---|---|
| | | | | VCOOUT × 1/4 (PREMIUM, ADVANCED) | VCOOUT× 1/5 (PREMIUM, ADVANCED) | VCOOUT × 1/6 (PREMIUM, ADVANCED, ECO) | |
| 8 (HS IntOSC)*2 | $00_B$ (Mr = 1) | $3B_H$ (Nr = 60) | 480.0 | N/A | N/A | 80.0*2 | 80.0*2 |
| | | $3A_H$ (Nr = 59) | 472.0 | N/A | N/A | 78.7*2 | 78.7*2 |
| | | $39_H$ (Nr = 58) | 464.0 | N/A | N/A | 77.3*2 | 77.3*2 |
| | | $38_H$ (Nr = 57) | 456.0 | N/A | N/A | 76.0*2 | 76.0*2 |
| | | $37_H$ (Nr = 56) | 448.0 | N/A | N/A | 74.7*2 | 74.7*2 |
| | | $36_H$ (Nr = 55) | 440.0 | N/A | N/A | 73.3*2 | 73.3*2 |
| | | $35_H$ (Nr = 54) | 432.0 | N/A | N/A | 72.0*2 | 72.0*2 |
| | | $34_H$ (Nr = 53) | 424.0 | N/A | N/A | 70.7*2 | 70.7*2 |
| | | $33_H$ (Nr = 52) | 416.0 | N/A | N/A | 69.3*2 | 69.3*2 |
| | | $32_H$ (Nr = 51) | 408.0 | N/A | N/A | 68.0*2 | 68.0*2 |
| | | $31_H$ (Nr = 50) | 400.0 | N/A | N/A | 66.7*2 | 66.7*2 |
| | | $30_H$ (Nr = 49) | 392.0 | N/A | N/A | 65.3*2 | 65.3*2 |
| | | $2F_H$ (Nr = 48) | 384.0 | N/A | N/A | 64.0*2 | 64.0*2 |

Note 1. The CPLLOUT frequency is defined by CKSC_CPUCLKD_CTL.CPUCLKDPLL[1:0]. Refer to the CKSC_CPUCLKD_CTL register description.

Note 2. Typical frequencies. User calibration of HS IntOSC is required before setting HS IntOSC as PLLCLKIN.

### 12.4.2.16 PLL Input Clock Selection

#### (1) CKSC_PLLIS_CTL — PLL Input Clock Selection Register

The correct write sequence using the PROTCMD1 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register can only be written, if the PLL is disabled.

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 A700$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PLLISCSID [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.18 CKSC_PLLIS_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | PLLISCSID [1:0] | Source Clock Setting for PLL input clock<br>$01_B$: MainOSC (Default)<br>$10_B$: HS IntOSC*1<br>Other than above: Setting prohibited |

Note 1. The maximum frequency of CPLLOUT and PPLLOUT is limited when the HS IntOSC is selected as the PLL input clock.

**CAUTION**

Set this register when PLL is stopped.

### (2) CKSC_PLLIS_ACT — PLL Input Clock Active Register

This register is initialized by all reset sources (ISORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 A708$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PLLISACT[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.19 CKSC_PLLIS_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1, 0 | PLLISACT[1:0] | Source clock for currently active PLL input clock |

### 12.4.2.17 PPLLCLK Source Clock Selection

#### (1) CKSC_PPLLCLKS_CTL — PPLLCLK Source Clock Selection Register

The correct write sequence using the PROTCMD1 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 A010$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PPLLCLKSCSID [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.20 CKSC_PPLLCLKS_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | PPLLCLKSCSID [1:0] | Source Clock Setting for PPLLCLK<br>$00_B$: Setting prohibited<br>$01_B$: EMCLK (default)<br>$10_B$: MainOSC<br>$11_B$: PPLLOUT |

### (2) CKSC_PPLLCLKS_ACT — PPLLCLK Source Clock Active Register

This register is initialized by all reset sources (ISORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 A018$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PPLLCLKSACT [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.21 CKSC_PPLLCLKS_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1, 0 | PPLLCLKSACT [1:0] | Source clock for currently active PPLLCLK*[1] |

Note 1. The data read from this register is $00_B$ if the selected source clock for all of the following clock domains is other than PPLLCLK (or PPLLCLK2):
C_ISO_PERI1, C_ISO_PERI2, C_ISO_LIN, C_ISO_ADCA, C_ISO_CAN, C_ISO_CSI, C_ISO_IIC

### 12.4.2.18 ROSCUT — HS IntOSC User Trimming Register

The correct write sequence using the PROTCMD0 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by the power-up reset signal PURES and $\overline{\text{CLMA0RES}}$.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 101C$_H$

**Value after reset:** 0000 0XXX$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | FADJUST[8:0] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 12.22 ROSCUT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 8 to 0 | FADJUST[8:0] | Frequency Adjustment parameters of HS IntOSC.<br>The value after reset of this register is pre-set value of HS IntOSC trimming data. Overwrite this register with “read value + 1” or “read value – 1” until the HS IntOSC frequency becomes target frequency range. |

## 12.4.3 Clock Selector Control Register

### 12.4.3.1 WDTA0 Clock Domain C_AWO_WDTA

#### (1) CKSC_AWDTAD_CTL — C_AWO_WDTA Clock Divider Selection Register

The correct write sequence using the PROTCMD0 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 2000$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | AWDTADCSID [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.23 CKSC_AWDTAD_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | AWDTADCSID [1:0] | Clock Divider Setting for C_AWO_WDTA<br>$00_B$: Setting prohibited<br>$01_B$: LS IntOSC / 128 (default)<br>$10_B$: LS IntOSC / 1<br>$11_B$: Setting prohibited |

**CAUTION**

Confirm that CKSC_AWDTAD_CTL is CKSC_AWDTAD_ACT before setting the CKSC_AWDTAD_CTL register.

### (2) CKSC_AWDTAD_ACT — C_AWO_WDTA Clock Divider Active Register

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 2008$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | AWDTADACT [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0*1 | 0*1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. The value in the inactive state. When the value becomes 01$_B$ after entering the active state.

**Table 12.24 CKSC_AWDTAD_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1, 0 | AWDTADACT [1:0] | Clock divider for currently active C_AWO_WDTA |

### (3) CKSC_AWDTAD_STPM — C_AWO_WDTA Stop Mask Register

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 2018$_H$

**Value after reset:** 0000 0002$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | AWDTADSTPMSK |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**CAUTION**

Do not change the “1” value after reset of bit 1.

**Table 12.25 CKSC_AWDTAD_STPM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | AWDTADSTPMSK*1 | 0: Clock domain C_AWO_WDTA is stopped in stand-by mode.<br>1: Clock domain C_AWO_WDTA is not stopped in stand-by mode. |

Note 1. The mode transition time can be optimized by setting 1 to this bit.

### 12.4.3.2 TAUJ Clock Domain C_AWO_TAUJ

#### (1) CKSC_ATAUJS_CTL — C_AWO_TAUJ Source Clock Selection Register

The correct write sequence using the PROTCMD0 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 $2100_H$

**Value after reset:** 0000 $0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | ATAUJSCSID[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 12.26 CKSC_ATAUJS_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 to 0 | ATAUJSCSID [2:0] | Source Clock Setting for C_AWO_TAUJ*1<br>$000_B$: Disabled<br>$001_B$: HS IntOSC (default)<br>$010_B$: MainOSC<br>$011_B$: LS IntOSC<br>$100_B$: PPLLCLK2<br>Other than above: Setting prohibited |

Note 1. Before transitioning to stand-by mode, select a source clock other than PPLLCLK2.

### (2) CKSC_ATAUJS_ACT — C_AWO_TAUJ Source Clock Active Register

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 2108$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | ATAUJSACT[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.27 CKSC_ATAUJS_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 to 0 | ATAUJSACT [2:0] | Source clock for currently active C_AWO_TAUJ |

### (3) CKSC_ATAUJD_CTL — C_AWO_TAUJ Clock Divider Selection Register

The correct write sequence using the PROTCMD0 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 2200$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | ATAUJDCSID[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 12.28 CKSC_ATAUJD_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 to 0 | ATAUJDCSID [2:0] | Clock Divider Setting for C_AWO_TAUJ<br>000$_B$: Setting prohibited<br>001$_B$: CKSC_ATAUJS_CTL selection /1 (default)<br>010$_B$: CKSC_ATAUJS_CTL selection /2<br>011$_B$: CKSC_ATAUJS_CTL selection /4<br>100$_B$: CKSC_ATAUJS_CTL selection /8<br>Other than above: Setting prohibited |

### (4) CKSC_ATAUJD_ACT — C_AWO_TAUJ Clock Divider Active Register

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 2208$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | ATAUJDACT[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.29 CKSC_ATAUJD_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 to 0 | ATAUJDACT [2:0] | Clock divider for currently active C_AWO_TAUJ |

### (5) CKSC_ATAUJD_STPM — C_AWO_TAUJ Stop Mask Register

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 2218$_H$

**Value after reset:** 0000 0002$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ATAUJ DSTP MSK |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**CAUTION**

Do not change the “1” value after reset of bit 1.

**Table 12.30 CKSC_ATAUJD_STPM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | ATAUJD STPMSK | 0: Clock domain C_AWO_TAUJ is stopped in stand-by mode.<br>1: Clock domain C_AWO_TAUJ is not stopped in stand-by mode. |

### 12.4.3.3 RTCA Clock Domain C_AWO_RTCA

#### (1) CKSC_ARTCAS_CTL — C_AWO_RTCA Source Clock Selection Register

The correct write sequence using the PROTCMD0 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by a power-up reset PURES.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 2300$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ARTCASCSID [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.31 CKSC_ARTCAS_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | ARTCASCSID [1:0] | Source Clock Setting for C_AWO_RTCA<br>$00_B$: Disable (default)<br>$01_B$: SubOSC*1<br>$10_B$: MainOSC*2<br>$11_B$: LS IntOSC |

Note 1. Supported only by the 144 pin and 176 pin products.

Note 2. To avoid supplying a clock signal equal to or higher than 4 MHz to the C_AWO_RTCA clock domain, check that CKSC_ARTCAD_ACT = 0000 0000$_H$ (disabled) before setting CKSC_ARTCAS_CTL to $10_B$ (MainOSC).

### (2) CKSC_ARTCAS_ACT — C_AWO_RTCA Source Clock Active Register

This register is initialized by a power-up reset PURES.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 2308$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ARTCASACT [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.32 CKSC_ARTCAS_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1, 0 | ARTCASACT [1:0] | Source clock for currently active C_AWO_RTCA |

### (3) CKSC_ARTCAD_CTL — C_AWO_RTCA Clock Divider Register

The correct write sequence using the PROTCMD0 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by a power-up reset PURES.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 2400$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | ARTCADCSID[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 12.33 CKSC_ARTCAD_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 to 0 | ARTCADCSID [2:0] | Clock Divider Setting for C_AWO_RTCA<br>$000_B$: Disabled (default)<br>$001_B$: CKSC_ARTCAS_CTL selection /1<br>$010_B$: CKSC_ARTCAS_CTL selection /2<br>$011_B$: CKSC_ARTCAS_CTL selection /4<br>$100_B$: CKSC_ARTCAS_CTL selection /8<br>Other than above: Setting prohibited |

### (4) CKSC_ARTCAD_ACT — C_AWO_RTCA Clock Divider Active Register

This register is initialized by a power-up reset PURES.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 2408$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | ARTCADACT[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.34 CKSC_ARTCAD_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 to 0 | ARTCADACT [2:0] | Clock divider for currently active C_AWO_RTCA |

### (5) CKSC_ARTCAD_STPM — C_AWO_RTCA Stop Mask Register

This register is initialized by a power-up reset PURES.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 2418$_H$

**Value after reset:** 0000 0002$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ARTCADSTPMSK |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**CAUTION**

Do not change the “1” value after reset of bit 1.

**Table 12.35 CKSC_ARTCAD_STPM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | ARTCAD STPMSK | 0: Clock domain C_AWO_RTCA is stopped in stand-by mode.<br>1: Clock domain C_AWO_RTCA is not stopped in stand-by mode. |

### 12.4.3.4 ADCA0 Clock Domain C_AWO_ADCA

#### (1) CKSC_AADCAS_CTL — C_AWO_ADCA Source Clock Selection Register

The correct write sequence using the PROTCMD0 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 $2500_H$

**Value after reset:** 0000 $0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | AADCASCSID [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.36 CKSC_AADCAS_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | AADCASCSID [1:0] | Source Clock Setting for C_AWO_ADCA*1<br>$00_B$: Disabled<br>$01_B$: HS IntOSC (default)<br>$10_B$: MainOSC<br>$11_B$: PPLLCLK2 |

Note 1. Before transitioning to stand-by mode, select a source clock other than PPLLCLK2.

### (2) CKSC_AADCAS_ACT — C_AWO_ADCA Source Clock Active Register

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 2508$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | AADCASACT [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.37 CKSC_AADCAS_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1, 0 | AADCASACT [1:0] | Source clock for currently active C_AWO_ADCA |

### (3) CKSC_AADCAD_CTL — C_AWO_ADCA Clock Divider Selection Register

The correct write sequence using the PROTCMD0 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 2600$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | AADCADCSID [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.38 CKSC_AADCAD_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | AADCADCSID [1:0] | Clock Divider Setting for C_AWO_ADCA<br>$00_B$: Setting prohibited<br>$01_B$: CKSC_AADCAS_CTL selection /1 (default)<br>$10_B$: CKSC_AADCAS_CTL selection /2*1<br>$11_B$: Setting prohibited |

Note 1. Make sure that the frequency of CKSC_AADCA is no less than 8 MHz after division by 2.

### (4) CKSC_AADCAD_ACT — C_AWO_ADCA Clock Divider Active Register

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 2608$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | AADCADACT [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.39 CKSC_AADCAD_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1, 0 | AADCADACT [1:0] | Clock divider for currently active for C_AWO_ADCA |

### (5) CKSC_AADCAD_STPM — C_AWO_ADCA Stop Mask Register

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 2618$_H$

**Value after reset:** 0000 0002$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | AADCA DSTP MSK |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**CAUTION**

Do not change the “1” value after reset of bit 1.

**Table 12.40 CKSC_AADCAD_STPM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | AADCAD STPMSK | 0: Clock domain C_AWO_ADCA is stopped in stand-by mode.<br>1: Clock domain C_AWO_ADCA is not stopped in stand-by mode. |

### 12.4.3.5 FOUT Clock Domain C_AWO_FOUT

#### (1) CKSC_AFOUTS_CTL — C_AWO_FOUT Source Clock Selection Register

The correct write sequence using the PROTCMD0 register is required in order to update this register For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 2700$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | AFOUTSCSID[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 12.41 CKSC_AFOUTS_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 to 0 | AFOUTSCSID [2:0] | Source Clock Setting for C_AWO_FOUT*1<br>$000_B$: Disabled (default)<br>$001_B$: MainOSC<br>$010_B$: HS IntOSC<br>$011_B$: LS IntOSC<br>$100_B$: SubOSC*2<br>$101_B$: PPLLCLK4<br>$110_B$: PPLLCLK4<br>$111_B$: Setting prohibited |

Note 1. Before transitioning to stand-by mode, select a source clock other than PPLLCLK4.

Note 2. Supported only by the 144 pin and 176 pin products.

### (2) CKSC_AFOUTS_ACT — C_AWO_FOUT Source Clock Active Register

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 2708$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | AFOUTSACT[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.42 CKSC_AFOUTS_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 to 0 | AFOUTSACT [2:0] | Source clock for currently active C_AWO_FOUT |

### (3) CKSC_AFOUTS_STPM — C_AWO_FOUT Stop Mask Register

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 2718$_H$

**Value after reset:** 0000 0002$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | AFOUTSSTPMSK |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**CAUTION**

Do not change the “1” value after reset of bit 1.

**Table 12.43 CKSC_AFOUTS_STPM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | AFOUTS STPMSK | 0: Clock domain C_AWO_FOUT is stopped in stand-by mode.<br>1: Clock domain C_AWO_FOUT is not stopped in stand-by mode. |

### 12.4.3.6 CPU Clock Domain C_ISO_CPUCLK

#### (1) CKSC_CPUCLKS_CTL — C_ISO_CPUCLK Source Clock Selection Register

The correct write sequence using the PROTCMD1 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 A000$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CPUCLKSCSID [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.44 CKSC_CPUCLKS_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | CPUCLKSCSID [1:0] | Source Clock Setting for C_ISO_CPUCLK<br>$00_B$: Setting prohibited<br>$01_B$: EMCLK (default)<br>$10_B$: MainOSC<br>$11_B$: CPLLOUT |

**CAUTION**

The clock source selected for the C_ISO_CPUCLK clock domain should not be stopped by software.

### (2) CKSC_CPUCLKS_ACT — C_ISO_CPUCLK Source Clock Active Register

This register is initialized by all reset sources (ISORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 A008$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CPUCLKSACT [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.45 CKSC_CPUCLKS_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1, 0 | CPUCLKSACT [1:0] | Source clock for currently active C_ISO_CPUCLK |

### (3) CKSC_CPUCLKD_CTL — C_ISO_CPUCLK Clock Divider Selection Register

The correct write sequence using the PROTCMD1 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 A100$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | CPUCLKDPLL [1:0] | | CPUCLKDCSID [2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 12.46 CKSC_CPUCLKD_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4, 3 | CPUCLKDPLL [1:0] | Clock Divider CPLLDIV Setting<br>Specifies the CPLLDIV divisor, which determines maximum clock frequency of C_ISO_CPUCLK.<br><br>ADVANCED, PREMIUM<br>$00_B$: CPLLOUT = VCOOUT × 1/6 (80 MHz)<br>$01_B$: CPLLOUT = VCOOUT × 1/5 (96 MHz)<br>$10_B$: CPLLOUT = VCOOUT × 1/4 (120 MHz)<br>$11_B$: Setting prohibited<br><br>ECO<br>$00_B$: CPLLOUT = VCOOUT × 1/6 (80 MHz)<br>Other than above: Setting prohibited |
| 2 to 0 | CPUCLKDCSID [2:0] | Clock Divider Setting for C_ISO_CPUCLK<br>$000_B$: Setting prohibited<br>$001_B$: CKSC_CPUCLKS_CTL selection /1 (Default)<br>$010_B$: CKSC_CPUCLKS_CTL selection /2<br>$011_B$: CKSC_CPUCLKS_CTL selection /4<br>$100_B$: CKSC_CPUCLKS_CTL selection /8<br>Other than above: Setting prohibited |

### (4) CKSC_CPUCLKD_ACT — C_ISO_CPUCLK Clock Divider Active Register

This register is initialized by all reset sources (ISORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 A108$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | CPUCLKDPLLACT[1:0] | | CPUCLKDACT[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.47 CKSC_CPUCLKD_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | When read, the value after reset is returned. |
| 4, 3 | CPUCLKDPLLACT[1:0] | Clock divider for currently active CPLLDIV. |
| 2 to 0 | CPUCLKDACT[2:0] | Clock divider for currently active C_ISO_CPUCLK |

### 12.4.3.7 Peripheral Clock Domains C_ISO_PERI1 and C_ISO_PERI2

#### (1) CKSC_IPERI1S_CTL — C_ISO_PERI1 Source Clock Selection Register

The correct write sequence using the PROTCMD1 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 $A200_H$

**Value after reset:** 0000 $0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IPERI1SCSID [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.48 CKSC_IPERI1S_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | IPERI1SCSID [1:0] | Source Clock Setting for C_ISO_PERI1<br>$00_B$: Disabled<br>$01_B$: PPLLCLK (default)<br>$10_B$: PPLLCLK<br>$11_B$: Setting prohibited |

### (2) CKSC_IPERI1S_ACT — C_ISO_PERI1 Source Clock Active Register

This register is initialized by all reset sources (ISORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 A208$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IPERI1SACT [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.49 CKSC_IPERI1S_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1, 0 | IPERI1SACT [1:0] | Source clock for currently active C_ISO_PERI1 |

### (3) CKSC_IPERI2S_CTL — C_ISO_PERI2 Source Clock Selection Register

The correct write sequence using the PROTCMD1 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 A300$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IPERI2SCSID [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.50 CKSC_IPERI2S_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | IPERI2SCSID [1:0] | Source Clock Setting for C_ISO_PERI2<br>$00_B$: Disabled<br>$01_B$: PPLLCLK2 (default)<br>$10_B$: PPLLCLK2<br>$11_B$: Setting prohibited |

### (4) CKSC_IPERI2S_ACT — C_ISO_PERI2 Source Clock Active Register

This register is initialized by all reset sources (ISORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 A308$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IPERI2SACT [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.51 CKSC_IPERI2S_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1, 0 | IPERI2SACT [1:0] | Source clock for currently active C_ISO_PERI2 |

### 12.4.3.8 RLIN Clock Domains C_ISO_LIN

#### (1) CKSC_ILINS_CTL — C_ISO_LIN Source Clock Selection Register

The correct write sequence using the PROTCMD1 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 A400$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | ILINSCSID[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 12.52 CKSC_ILINS_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 to 0 | ILINSCSID[2:0] | Source Clock Setting for C_ISO_LIN*1<br>$000_B$: Disabled<br>$001_B$: PPLLCLK2 (default)<br>$010_B$: MainOSC<br>$011_B$: PPLLCLK2<br>$100_B$: HS IntOSC<br>Other than above: Setting prohibited |

Note 1. Before transitioning to stand-by mode, select a source clock other than PPLLCLK2.

### (2) CKSC_ILINS_ACT — C_ISO_LIN Source Clock Active Register

This register is initialized by all reset sources (ISORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 A408$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | ILINSACT[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.53 CKSC_ILINS_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 to 0 | ILINSACT[2:0] | Source clock for currently active C_ISO_LIN |

### (3) CKSC_ILIND_CTL — C_ISO_LIN Clock Divider Selection Register

The correct write sequence using the PROTCMD1 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 A800$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ILINDCSID[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.54 CKSC_ILIND_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | ILINDCSID[1:0] | Clock Divider Setting for C_ISO_LIN<br>$00_B$: Setting prohibited<br>$01_B$: CKSC_ILINS_CTL selection /1 (default)<br>$10_B$: CKSC_ILINS_CTL selection /4<br>$11_B$: CKSC_ILINS_CTL selection /8 |

**NOTE**

The setting of this register is only applicable to RLIN30. The settings $10_B$ (CKSC_ILINS_CTL selection / 4) and $11_B$ (CKSC_ILINS_CTL selection /8) are only available in UART mode.

### (4) CKSC_ILIND_ACT — C_ISO_LIN Clock Divider Active Register

This register is initialized by all reset sources (ISORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 A808$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ILINDACT[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.55 CKSC_ILIND_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1, 0 | ILINDACT[1:0] | Clock divider for currently active C_ISO_LIN |

### (5) CKSC_ILIND_STPM — C_ISO_LIN Stop Mask Register

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 A818$_H$

**Value after reset:** 0000 0002$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ILIND STP MSK |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**CAUTION**

Do not change the “1” value after reset of bit 1.

**Table 12.56 CKSC_ILIND_STPM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | ILINDSTPMSK | 0: Clock domain C_ISO_LIN is stopped in stand-by mode.<br>1: Clock domain C_ISO_LIN is not stopped in stand-by mode. |

### 12.4.3.9 ADCA1 Clock Domain C_ISO_ADCA

#### (1) CKSC_IADCAS_CTL — C_ISO_ADCA Source Clock Selection Register

The correct write sequence using the PROTCMD1 register is required in order to update this register For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 A500$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IADCASCSID [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.57 CKSC_IADCAS_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | IADCASCSID [1:0] | Source Clock Setting for C_ISO_ADCA<br>$00_B$: Disabled<br>$01_B$: HS IntOSC (default)<br>$10_B$: MainOSC<br>$11_B$: PPLLCLK2 |

**CAUTION**

The CKSC_IADCAS_CTL register and the CKSC_IADCAD_CTL register must be set so that the relationship between frequency (1) and (2) is retained within the range of “(1) / (2) = 2 to 4.8”.

(1) Frequency [MHz] specified by the C_ISO_CPUCLK source clock selection register (CKSC_CPUCLKS_CTL) and C_ISO_CPUCLK clock divider selection register (CKSC_CPUCLKD_CTL)

(2) Frequency [MHz] specified by the C_ISO_ADCA source clock selection register (CKSC_IADCAS_CTL) and the C_ISO_ADCA clock divider selection register (CKSC_IADCAD_CTL)

### (2) CKSC_IADCAS_ACT — C_ISO_ADCA Source Clock Active Register

This register is initialized by all reset sources (ISORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 A508$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IADCASACT [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.58 CKSC_IADCAS_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1, 0 | IADCASACT [1:0] | Source clock for currently active C_ISO_ADCA |

### (3) CKSC_IADCAD_CTL — C_ISO_ADCA Clock Divider Selection Register

The correct write sequence using the PROTCMD1 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 A600$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IADCADCSID [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.59 CKSC_IADCAD_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | IADCADCSID [1:0] | Clock Divider Setting for C_ISO_ADCA<br>$00_B$: Setting prohibited<br>$01_B$: CKSC_IADCAS_CTL selection /1 (default)<br>$10_B$: CKSC_IADCAS_CTL selection /2*1<br>$11_B$: Setting prohibited |

Note 1. Make sure that the frequency of CKSC_IADCA is no less than 8 MHz after division by 2.

**CAUTION**

The CKSC_IADCAS_CTL register and the CKSC_IADCAD_CTL register must be set so that the relationship between frequency (1) and (2) is retained within the range of "(1) / (2) = 2 to 4.8".

(1) Frequency [MHz] specified by the C_ISO_CPUCLK source clock selection register (CKSC_CPUCLKS_CTL) and C_ISO_CPUCLK clock divider selection register (CKSC_CPUCLKD_CTL)

(2) Frequency [MHz] specified by the C_ISO_ADCA source clock selection register (CKSC_IADCAS_CTL) and the C_ISO_ADCA clock divider selection register (CKSC_IADCAD_CTL)

### (4) CKSC_IADCAD_ACT — C_ISO_ADCA Clock Divider Active Register

This register is initialized by all reset sources (ISORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 A608$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IADCADACT [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.60 CKSC_IADCAD_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1, 0 | IADCADACT [1:0] | Clock divider for currently active C_ISO_ADCA |

### 12.4.3.10 RS-CAN Clock Domains C_ISO_CAN and C_ISO_CANOSC

#### (1) CKSC_ICANS_CTL — C_ISO_CAN Source Clock Selection Register

The correct write sequence using the PROTCMD1 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 A900$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ICANSCSID[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.61 CKSC_ICANS_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | ICANSCSID[1:0] | Source Clock Setting for C_ISO_CAN*1<br>$00_B$: Disabled<br>$01_B$: MainOSC<br>$10_B$: PPLLCLK<br>$11_B$: PPLLCLK (default) |

Note 1. Before transitioning to stand-by mode, select a source clock other than PPLLCLK.

### (2) CKSC_ICANS_ACT — C_ISO_CAN Source Clock Active Register

This register is initialized by all reset sources (ISORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 A908$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ICANSACT[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.62 CKSC_ICANS_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1, 0 | ICANSACT[1:0] | Source clock for currently active C_ISO_CAN |

### (3) CKSC_ICANS_STPM — C_ISO_CAN Stop Mask Register

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 A918$_H$

**Value after reset:** 0000 0002$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ICANS STP MSK |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**CAUTION**

Do not change the “1” value after reset of bit 1.

**Table 12.63 CKSC_ICANS_STPM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | ICANS STPMSK | 0: Clock domain C_ISO_CAN is stopped in stand-by mode.<br>1: Clock domain C_ISO_CAN is not stopped in stand-by mode. |

### (4) CKSC_ICANOSCD_CTL — C_ISO_CANOSC Clock Divider Selection Register

The correct write sequence using the PROTCMD1 register is required in order to update this register For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 AA00$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ICANOSCD CSID[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.64 CKSC_ICANOSCD_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | ICANOSCD CSID[1:0] | Clock Divider Setting for C_ISO_CANOSC*1<br>$00_B$: Disabled (default)<br>$01_B$: MainOSC/1<br>$10_B$: MainOSC/2<br>$11_B$: Setting prohibited |

Note 1. Select MainOSC/2 when the source clock setting of C_ISO_CAN is MainOSC.

### (5) CKSC_ICANOSCD_ACT — C_ISO_CANOSC Clock Divider Active Register

This register is initialized by all reset sources (ISORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 AA08$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ICANOSCDACT [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.65 CKSC_ICANOSCD_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1, 0 | ICANOSCDACT [1:0] | Clock divider for currently active C_ISO_CANOSC |

### (6) CKSC_ICANOSCD_STPM — C_ISO_CANOSC Stop Mask Register

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 AA18$_H$

**Value after reset:** 0000 0002$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ICANOSCDSTPMSK |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**CAUTION**

Do not change the “1” value after reset of bit 1.

**Table 12.66 CKSC_ICANOSCD_STPM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | ICANOSCD STPMSK | 0: Clock domain C_ISO_CANOSC is stopped in stand-by mode.<br>1: Clock domain C_ISO_CANOSC is not stopped in stand-by mode. |

### 12.4.3.11 CSI Clock Domain C_ISO_CSI

#### (1) CKSC_ICSIS_CTL — C_ISO_CSI Source Clock Selection Register

The correct write sequence using the PROTCMD1 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 AB00$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | ICSISCSID[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 12.67 CKSC_ICSIS_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 to 0 | ICSISCSID[2:0] | Source Clock Setting for C_ISO_CSI<br>$000_B$: Disabled<br>$001_B$: PPLLCLK (default)<br>$010_B$: PPLLCLK<br>$011_B$: MainOSC<br>$100_B$: HS IntOSC<br>Other than above: Setting prohibited |

### (2) CKSC_ICSIS_ACT — C_ISO_CSI Source Clock Active Register

This register is initialized by all reset sources (ISORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 AB08$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | ICSISACT[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.68 CKSC_ICSIS_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 to 0 | ICSISACT[2:0] | Source clock for currently active C_ISO_CSI |

### 12.4.3.12 IIC Clock Domain C_ISO_IIC

#### (1) CKSC_IIICS_CTL — C_ISO_IIC Source Clock Selection Register

The correct write sequence using the PROTCMD1 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources (ISORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 AC00$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IIICSCSID[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 12.69 CKSC_IIICS_CTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | IIICSCSID[1:0] | Source Clock Setting for C_ISO_IIC<br>$00_B$: Disabled<br>$01_B$: PPLLCLK2 (default)<br>$10_B$: PPLLCLK2<br>$11_B$: Setting prohibited |

### (2) CKSC_IIICS_ACT — C_ISO_IIC Source Clock Active Register

This register is initialized by all reset sources (ISORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 AC08$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IIICSACT[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.70 CKSC_IIICS_ACT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1, 0 | IIICSACT[1:0] | Source clock for currently active C_ISO_IIC |

# 12.5 Clock Domain Setting Method

## 12.5.1 Clock Domain Setting

### 12.5.1.1 Overview of Clock Selector Register

The clock selector for a clock domain C_AWO_<name>/C_ISO_<name> can be controlled by the following registers:

- Source clock selection registers
  These registers select the clock to be used as the domain clock from the available source clocks.
  - AWO source clock selection: CKSC_A<name>S_CTL
  - ISO source clock selection: CKSC_I<name>S_CTL
- Clock divider selection registers
  These registers specify the clock division ratio for the selected source clock.
  - AWO clock divider: CKSC_A<name>D_CTL
  - ISO clock divider: CKSC_I<name>D_CTL
- Source clock active registers and clock divider active registers
  These registers return the currently active source clock selection and division ratio, respectively.
  - AWO source clock active register/clock divider active register: CKSC_A<name>S_ACT/ CKSC_A<name>D_ACT
  - ISO source clock active register/clock divider active register: CKSC_I<name>S_ACT/ CKSC_I<name>D_ACT

**NOTE**

- Not all clock selectors provide all the control functions described above.
- The symbol “I”, which indicates the power domain, is not added to the names of registers within clock domain C_ISO_CPUCLK.

#### 12.5.1.2 Setting Procedure for Clock Domain

Procedure of setting up clock domain is described as below:

1. Set up a source clock
   - Select a source clock. (CKSC_A<name>S_CTL, CKSC_I<name>S_CTL)
   - Confirm completion of selection. (CKSC_A<name>S_ACT, CKSC_I<name>S_ACT)*1
2. Setting a clock divider
   - Select a clock divider. (CKSC_A<name>D_CTL, CKSC_I<name>D_CTL)
   - Confirm completion of selection. (CKSC_A<name>D_ACT, CKSC_I<name>D_ACT)*2

**Note 1.** Continue processing after CKSC_A<name>S_ACT and CKSC_I<name>S_ACT are updated with the new values written to CKSC_A<name>S_CTL and CKSC_I<name>S_CTL.

**Note 2.** Continue processing after CKSC_A<name>D_ACT and CKSC_I<name>D_ACT are updated with the new values written to CKSC_A<name>D_CTL and CKSC_I<name>D_CTL.

**CAUTION**

**The source clock to be selected must be operating before performing these settings.**

**The behavior and performance are not guaranteed if setup is performed while the source clock is stopped.**

**Access to a peripheral module is prohibited while the clock is not supplied to the module.**

### 12.5.2 Stopping the Clock in Stand-by Mode

In stand-by mode (STOP mode, DeepSTOP mode, and Cyclic STOP mode), clock domain C_AWO_<name>/C_ISO_<name> can be configured to stop or continue its clock CKSCLK_A<name>/CKSCLK_I<name> in response to clock stop requests from the stand-by controller.

The clock stop mask registers are used to determine the operation status of the clock in stand-by mode:

- CKSC_A<name>_STPM.A<name>STPMSK/CKSC_I<name>_STPM.I<name>STPMSK = 0:
  The STOP request signal is not masked, so the domain clock CKSCLK_A<name>/CKSCLK_I<name> is stopped during stand-by mode.
  If the domain clock was in operation before transition to stand-by mode, it is automatically re-started after wake-up from stand-by mode.
  If there is another clock domain for which the same source clock is selected and its stop mask setting is set to 1 (CKSC_A<name>_STPM.A<name>STPMSK/CKSC_I<name>_STPM.I<name>STPMSK=1), the source clock will continue operation in stand-by mode.
  The CPU clock domain C_ISO_CPUCLK is always stopped in stand-by mode.
- CKSC_A<name>_STPM.A<name>STPMSK/CKSC_I<name>_STPM.I<name>STPMSK = 1:
  The STOP request signal is masked, so CKSCLK_A<name>/CKSCLK_I<name> continues to operate during stand-by.
  The source clock selected for the target clock domain will also continue to operate in stand-by mode.
  Supply of a clock signal to the clock domains in the Isolated area will be stopped in DeepSTOP mode.

### 12.5.3 Clock Domain Settings

The following table shows a selectable source clock, a frequency division ratio, and a register to be used for each clock domain.

**Table 12.71 List of Selectable Clocks (1/2)**

| Clock Domain | Clock Name | Selectable Register | | Frequency Divided Register | | Maximum Frequency | Applicable Unit |
|---|---|---|---|---|---|---|---|
| C_AWO_WDTA | CKSCLK_AWDTA | — | **LS IntOSC** | CKSC_AWDTAD_CTL | 1/1 | 240 kHz | WDTA0 |
| | | | | | **1/128** | | |
| C_AWO_TAUJ | CKSCLK_ATAUJ | CKSC_ATAUJS_CTL | MainOSC | CKSC_ATAUJD_CTL | **1/1** | 40 MHz | TAUJ0 |
| | | | **HS IntOSC** | | 1/2 | | |
| | | | LS IntOSC | | 1/4 | | |
| | | | PPLLCLK2 | | 1/8 | | |
| | | | Disable | | — | | |
| C_AWO_RTCA | CKSCLK_ARTCA | CKSC_ARTCAS_CTL | MainOSC | CKSC_ARTCAD_CTL | 1/1 | 4 MHz | RTCA0 |
| | | | LS IntOSC | | 1/2 | | |
| | | | SubOSC*1 | | 1/4 | | |
| | | | **Disable** | | 1/8 | | |
| | | | — | | **Disable** | | |
| C_AWO_ADCA | CKSCLK_AADCA | CKSC_AADCAS_CTL | MainOSC | CKSC_AADCAD_CTL | **1/1** | 40 MHz | ADCA0 |
| | | | **HS IntOSC** | | 1/2 | | |
| | | | PPLLCLK2 | | **—** | | |
| | | | Disable | | | | |
| C_AWO_FOUT | CKSCLK_AFOUT | CKSC_AFOUTS_CTL | MainOSC | — | **1/1** | 24 MHz | FOUT |
| | | | HS IntOSC | | | | |
| | | | LS IntOSC | | | | |
| | | | SubOSC*1 | | | | |
| | | | PPLLCLK4 | | | | |
| | | | **Disable** | | | | |
| C_ISO_CPUCLK | CPUCLK | CKSC_CPUCLKS_CTL *5 | MainOSC | CKSC_CPUCLKD_CTL *5 | **1/1** | 120 / 80 MHz*3 | CPU subsystem |
| | | | CPLLOUT (VCOOUT × 1/4)*2 | | 1/2 | | |
| | | | CPLLOUT (VCOOUT × 1/5)*2 | | 1/4 | | |
| | | | CPLLOUT (VCOOUT × 1/6) | | 1/8 | | |
| | | | **EMCLK** | | **—** | | |
| C_ISO_PERI1 | CKSCLK_IPERI1 | CKSC_IPERI1S_CTL | **PPLLCLK** | — | **1/1** | 80 MHz | TAUD0<br>TAUJ1<br>ENCA0<br>TAPA0<br>PIC0 |
| | | | Disable | | | | |
| C_ISO_PERI2 | CKSCLK_IPERI2 | CKSC_IPERI2S_CTL | **PPLLCLK2** | — | **1/1** | 40 MHz | TAUBn<br>PWBAn<br>PWGAn<br>PWSAn<br>RS-CANn (clkc) |
| | | | Disable | | | | |
| C_ISO_LIN | CKSCLK_ILIN | CKSC_ILINS_CTL | MainOSC | CKSC_ILIND_CTL*4 | **1/1** | 40 MHz | RLIN2m<br>RLIN3n |
| | | | HS IntOSC | | 1/4 | | |
| | | | **PPLLCLK2** | | 1/8 | | |
| | | | Disable | | | | |

Table 12.71 List of Selectable Clocks (2/2)

| Clock Domain | Clock Name | Selectable Register | | Frequency Divided Register | | Maximum Frequency | Applicable Unit |
|---|---|---|---|---|---|---|---|
| C_ISO_ADCA | CKSCLK_IADCA | CKSC_IADCAS_CTL | MainOSC | CKSC_IADCAD_CTL | **1/1** | 40MHz | ADCA1 |
| | | | **HS IntOSC** | | 1/2 | | |
| | | | PPLLCLK2 | | — | | |
| | | | Disable | | | | |
| C_ISO_CAN | CKSCLK_ICAN | CKSC_ICANS_CTL | MainOSC | — | **1/1** | 80 MHz | RS-CANn (pclk) |
| | | | **PPLLCLK** | | | | |
| | | | Disable | | | | |
| C_ISO_CANOSC | CKSCLK_ICANOSC | — | **MainOSC** | CKSC_ICANOSCD_CTL | 1/1 | 24 MHz | RS-CANn (clk_xincan) |
| | | | | | 1/2 | | |
| | | | | | **Disable** | | |
| C_ISO_CSI | CKSCLK_ICSI | CKSC_ICSIS_CTL | **PPLLCLK** | — | **1/1** | 80 MHz | CSIGn CSIHn |
| | | | MainOSC | | | | |
| | | | HS IntOSC | | | | |
| | | | Disable | | | | |
| C_ISO_IIC | CKSCLK_IIIC | CKSC_IIICS_CTL | **PPLLCLK2** | — | **1/1** | 40 MHz | RIICn |
| | | | Disable | | | | |

**Note:** The items written in **bold** are the initial setting clocks for each register.

Note 1. Supported only by 144 pin and 176 pin products.

Note 2. These settings are only supported by ADVANCED/PREMIUM products.

Note 3. Operation at 120MHz/96MHz is only supported by ADVANCED/PREMIUM products.

Note 4. The setting of this register only applies to RLIN30. The settings 1/4 and 1/8 are only available in UART mode.

Note 5. CKSC_CPUCLKS_CTL selects selection of MainOSC, CPLLOUT and EMCLK. CKSC_CPUCLKD_CTL.CPUCLKDPLL[1:0] selects CPLLOUT clock frequency.

**CAUTION**

**To stop the clock source selected for the clock domain before transitioning to STOP/ DeepSTOP mode, select “Disable” for that clock domain in advance. Do not stop the source clock of a clock domain for which “Disable” cannot be selected while functions are operating on that clock domain. To stop the clock source selected for the domain by transitioning to STOP/DeepSTOP mode, “Disable” does not need to be selected. Instead of setting “Disable”, select “Stop” for the clock domain in stand-by mode by using the stop mask register.**

# 12.6 Frequency Output Function (FOUT)

The frequency output function (FOUT) allows the clock to be output the clock as the external signal. Furthermore, the frequency can be divided by the clock divider before it is output.

## 12.6.1 Functional Overview

**Figure 12.9** shows the configuration of the frequency output function.

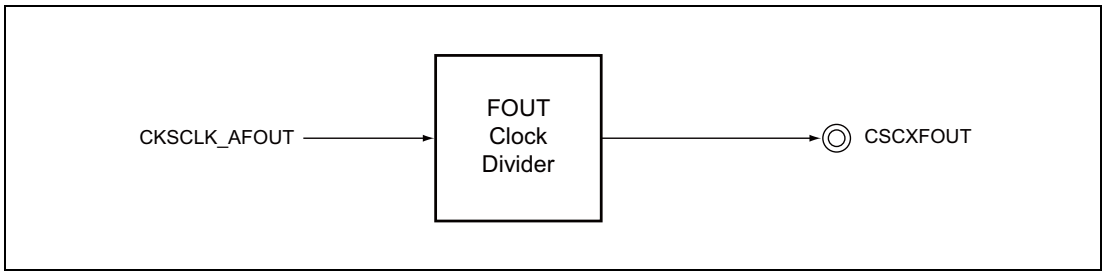



**Figure 12.9 Frequency Output Function**

The clock output function outputs the CKSCLK_AFOUT clock divided by 1 to 63 through the clock divider from CSCXFOUT. Division ratio N is set to the FOUTDIV[5:0] bits in the FOUTDIV register. Clock output frequency $f_{CSCXFOUT}$ is expressed by the following equation.

$$f_{CSCXFOUT} = (\text{CKSCLK_AFOUT clock frequency}) / N$$

Clock output starts when, after CKSCLK_AFOUT is set and the clock output for the pin function is selected, division ratio N is set to the FOUTDIV[5:0] bits in the FOUTDIV register.

When a new division ratio is written to the FOUTDIV.FOUTDIV[5:0] bits, it becomes effective in synchronization with the CSCXFOUT output clock. Accordingly, the division ratio can be changed even while the CSCXFOUT clock is operating. The clock output is stopped by writing $000_H$ to the FOUTDIV[5:0] bits.

## 12.6.2 Clock Supply

The clock supply to the CSCXFOUT is shown in the following table.

**Table 12.72 Clock Supply**

| Module | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| CSCXFOUT | PCLK | CPUCLK4 | Bus clock (Register access) |
| | CKSCLK_AFOUT | CKSCLK_AFOUT | Clock source of FOUT clock divider |

## 12.6.3 Registers

### 12.6.3.1 List of Registers

The FOUT registers are listed in the following table.

**Table 12.73 Register of the CSCXFOUT Clock Divider**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| CLKCTL | Clock divider register | FOUTDIV | FFF8 $2800_H$ |
| | Clock divider status register | FOUTSTAT | FFF8 $2804_H$ |

### 12.6.3.2 FOUTDIV — Clock Division Ratio Register

This register defines the clock divisor.

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 2800$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | FOUTDIV[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 12.74 FOUTDIV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 to 0 | FOUTDIV[5:0] | Clock Divider N<br>$00_H$: Clock output is stopped<br>$01_H$: N = 1<br>$02_H$: N = 2<br>...<br>$3E_H$: N = 62<br>$3F_H$: N = 63 |

### 12.6.3.3 FOUTSTAT — Clock Divider Status Register

This register indicates the clock output status.

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 2804$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | FOUTCLKACT | FOUTSYNC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.75 FOUTSTAT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1 | FOUTCLKACT | Clock Divider Active<br>0: Frequency output is stopped.<br>1: Frequency output is ongoing. |
| 0 | FOUTSYNC | Clock Divider Synchronized<br>0: The clock divider is in the process of synchronization.<br>1: The clock divider is stable (or stopped). |

# Section 13 Clock Monitor (CLMA)

This section describes clock monitor (CLMA).

The first section describes the attributes specific to the RH850/F1K microcontrollers, including the number of channels, register base addresses, and input/output signal names. The ensuing sections describe the functions relevant to all operations.

## 13.1 Features of RH850/F1K CLMA

### 13.1.1 Number of Channels

The RH850/F1K microcontrollers incorporate CLMA with the following number of channels.

**Table 13.1 Number of Channels**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of channels | | 3 | |
| Name | | CLMA0, CLMA1, CLMA2 | |

### 13.1.2 Register Base Addresses

The CLMA base addresses are listed in the following table.

The CLMA register addresses are expressed as an offset of the base address.

**Table 13.2 Register Base Addresses**

| Base Address Name | Base Address |
|---|---|
| <CLMA0_base> | FFF8 $C000_H$ |
| <CLMA1_base> | FFF8 $D000_H$ |
| <CLMA2_base> | FFF8 $E000_H$ |

### 13.1.3 Clock Supply to CLMA

The clocks monitored by CLMA and the CLMA sampling clocks are indicated below.

**Table 13.3 Clock Supply**

| Channel Name | Unit Clock Name | Supply Clock Name |
|---|---|---|
| CLMA0 | CLMATMON (monitored clock) | HS IntOSC |
| | CLMATSMP (sampling clock) | LS IntOSC |
| | Register access clock | CPUCLK4 |
| CLMA1 | CLMATMON (monitored clock) | MainOSC |
| | CLMATSMP (sampling clock) | LS IntOSC |
| | Register access clock | CPUCLK4 |
| CLMA2 | CLMATMON (monitored clock) | PPLLOUT |
| | CLMATSMP (sampling clock) | HS IntOSC |
| | Register access clock | CPUCLK4 |

### 13.1.4 Reset Sources

The reset sources of the CLMA are listed in the following table. The CLMA are initialized by these reset sources.

**Table 13.4 Reset Sources**

| Channel Name | Reset Source |
|---|---|
| CLMA0 | Reset sources other than transition to DeepSTOP mode (AWORES) |
| CLMA1 | Reset sources other than transition to DeepSTOP mode (AWORES) |
| CLMA2 | All reset sources (ISORES) |
| Common Registers (CLMATEST, CLMATESTS) | Power-up reset PURES (power-on-clear or debugger reset) |

### 13.1.5 Internal Input/Output Signals

The internal input/output signals of CLMA are listed in the following table.

**Table 13.5 Internal Input/Output Signals**

| Unit Signal Name | Description | Connection |
|---|---|---|
| $\overline{\text{CLMATRES}}$ | CLMA0 error reset output | Reset controller ($\overline{\text{CLMA0RES}}$) |
| $\overline{\text{CLMATRES}}$ | CLMA1 error reset output | Reset controller ($\overline{\text{CLMA1RES}}$) |
| $\overline{\text{CLMATRES}}$ | CLMA2 error reset output | Reset controller ($\overline{\text{CLMA2RES}}$) |

## 13.2 Overview

### 13.2.1 Functional Overview

Clock monitor CLMA detects frequency abnormalities in the monitored clock. It uses sampling clock CLMATSMP to monitor whether the frequency of input clock CLMATMON is within a specific range. Upon detection of an abnormal clock, it outputs a reset request signal.

The main components of the clock monitor are shown in **Figure 13.1**.

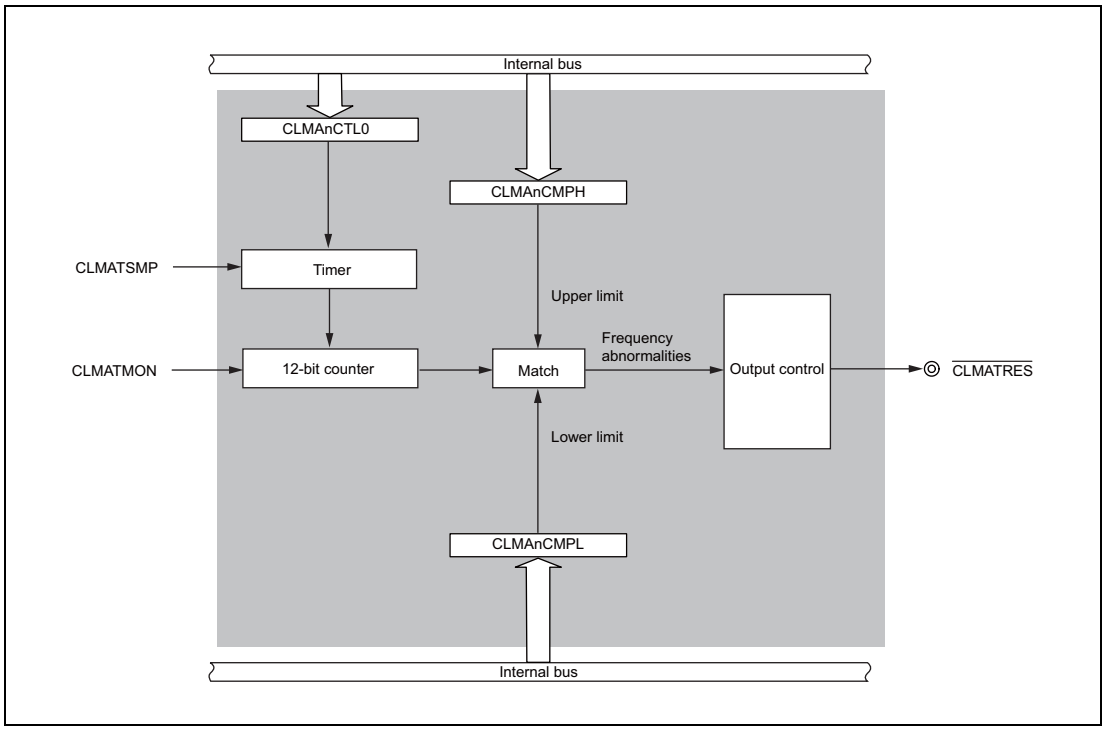



**Figure 13.1 Block Diagram of the Clock Monitor**

**NOTE**

Once enabled, only a reset can disable the CLMAn.

## 13.3 Enabling CLMA

Clock monitoring is started by the clock monitor when CLMAnCTL0.CLMAnCLME = 1.

When the monitored clock is stopped by a register operation or transition to stand-by mode, the corresponding clock monitor is automatically disabled. After the monitored clock starts oscillation again and becomes stable, the clock monitor also starts operation.

Since CLMA2 is initialized on return from DeepSTOP, the CLMA2 register must be set again before further operation is started.

## 13.4 Functions

### 13.4.1 Detection of Abnormal Clock Frequencies

#### Detection Method

1. CLMAn counts the rising edges of the monitored clock CLMATMON within 16 cycles of the sampling clock CLMATSMP and then compares the counter value with the specified thresholds:
   - CLMAnCMPL.CLMAnCMPL[11:0] defines the lower threshold.
   - CLMAnCMPH.CLMAnCMPH[11:0] defines the upper threshold.
2. When CLMATMON frequency*1 is lower than the limit, the counter falls below CLMAnCMPL.CLMAnCMPL[11:0].
3. When the frequency of CLMATMON is higher than the limit, the counter exceeds CLMAnCMPH.CLMAnCMPH[11:0].

**Note 1.** There is a case that the abnormal state is not detected when the monitor clock completely stops.

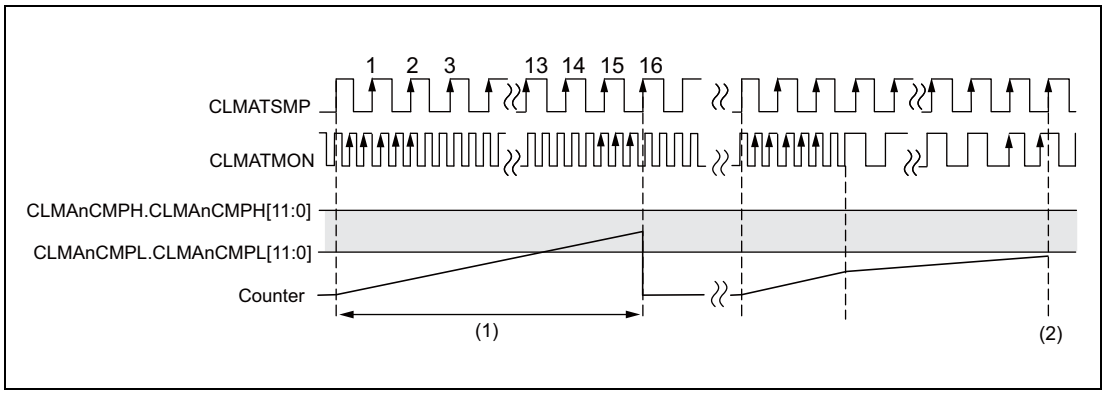



**Figure 13.2 Example: $f_{CLMATMON}$ is Lower than the Specified Limit**

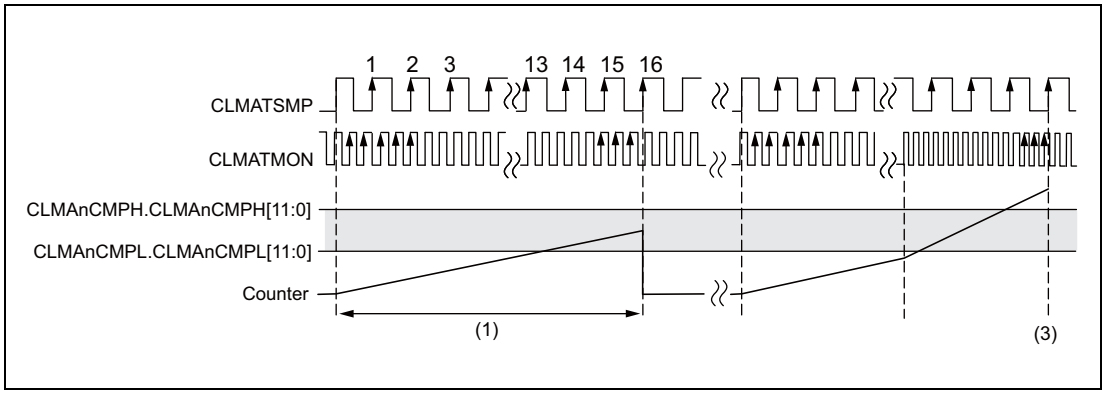



**Figure 13.3 Example: $f_{CLMATMON}$ is Higher than the Specified Limit**

**NOTE**

Even if $f_{CLMATMON}$ exceeds or falls below the specified limits during a sampling interval, the counter might be within the valid range.

Abnormal $f_{CLMATMON}$ is detected after one sampling interval.

**(1) Calculation method of the thresholds CLMAnCMPL.CLMAnCMPL[11:0] and CLMAnCMPH.CLMAnCMPH[11:0]**

The compare registers CLMAnCMPL and CLMAnCMPH are configured with the minimum and maximum number of clock cycles of CLMATMON that are assumed to be valid within 16 cycles of the sampling clock CLMATSMP.

The expected number of clock cycles is denoted by N.

$$\frac{16}{f_{CLMATSMP}} = \frac{N}{f_{CLMATMON}}$$

$$N = \frac{f_{CLMATMON}}{f_{CLMATSMP}} \times 16$$

Considering the allowed frequency deviations of CLMATMON and CLMATSMP, the threshold values can be calculated by the following formulas:

$$\text{Lower threshold} = N_{min} = \frac{f_{CLMATMON(dmin)}}{f_{CLMATSMP(max)}} \times 16 - 1$$

$$\text{Upper threshold} = N_{max} = \frac{f_{CLMATMON(max)}}{f_{CLMATSMP(min)}} \times 16 + 1$$

**NOTE**

The jitter of the PLL is covered by “+1” and “–1” in the formulas.

**Example:**

When $f_{CLMATSMP}$ = 240 kHz (±8%) and $f_{CLMATMON}$ = 16 MHz (±5%), the recommended threshold values are as follows:

$N_{min}$ = 15,200 / 259.2 × 16 – 1
= 937.27
CLMAnCMPL = 937 = $03A9_H$

$N_{max}$ = 16,800 / 220.8 × 16 + 1
= 1218.39
CLMAnCMPH = 1219 = $04C3_H$

**Minimum thresholds**

The following restrictions must be taken into account:

- CLMAnCMPL ≥ $0001_H$
- CLMAnCMPH ≥ CLMAnCMPL + $0003_H$

**(2) Definition of the initial value input to the threshold registers**

The reset values of the threshold registers are set so that the maximum frequency deviation of the monitored clock is allowed:

- CLMAnCMPL[11:0] = $001_H$
- CLMAnCMPH[11:0] = $3FF_H$

## 13.4.2 Notification of Abnormal Clock Frequency

If $f_{CLMATMON}$ exceeds the upper threshold or falls below the lower threshold, this is indicated as follows:

1. The reset request signal $\overline{\text{CLMATRES}}$ is set to low level.
2. The system reset (AWORES or ISORES) is generated and CLMAn is reset.

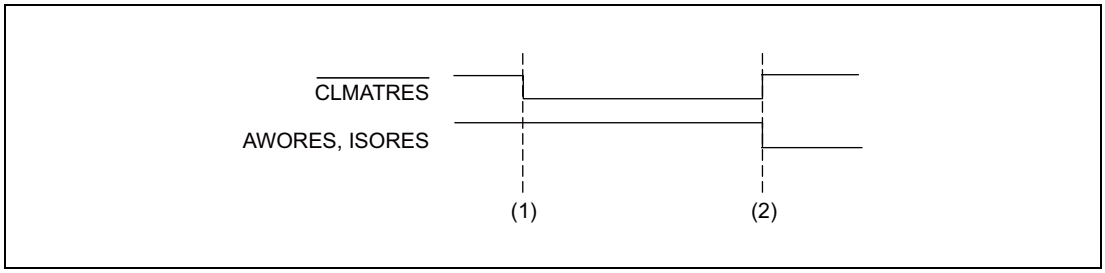



**Figure 13.4 Error Request Signal Output if $f_{CLMATMON}$ Exceeds Upper Threshold**

**CAUTION**

**For usage notes for CLMAn abnormality detection, see Section 13.6, Usage Notes for CLMAn.**

## 13.5 Registers

### 13.5.1 List of Registers

The following table lists the CLMA registers.

<CLMAn_base> is defined in **Section 13.1.2, Register Base Addresses**.

**Table 13.6 List of Clock Monitor Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| CLMAn | CLMAn control register 0 | CLMAnCTL0 | <CLMAn_base> + $00_H$ |
| | CLMAn compare register L | CLMAnCMPL | <CLMAn_base> + $08_H$ |
| | CLMAn compare register H | CLMAnCMPH | <CLMAn_base> + $0C_H$ |
| | CLMA test register | CLMATEST | FFF8 $C100_H$ |
| | CLMA test status register | CLMATESTS | FFF8 $C104_H$ |
| | CLMAn emulation register 0 | CLMAnEMU0 | <CLMAn_base> + $18_H$ |

### 13.5.2 CLMAnCTL0 — CLMAn Control Register 0

This register enables the clock monitor CLMAn.

The correct write sequence using the CLMAnPCMD register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

**Access:** This register can be read or written in 8-bit units.

**Address:** <CLMAn_base> + $00_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CLMAnCLME |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 13.7 CLMAnCTL0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | CLMAnCLME | Enables or disables the clock monitor.<br>0: CLMAn is disabled.<br>1: CLMAn is enabled. |

**CAUTION**

After the CLMAnCTL0.CLMAnCLME bit is set to 1, writing 0 to this bit is ignored. The only condition for clearing the bit is a reset (AWORES, ISORES). In addition, the bit is cleared when the CLMATEST.RESCLM bit is set to 1 during self-test of CLMAn.

### 13.5.3 CLMAnCMPH — CLMAn Compare Register H

This register specifies the upper limit of frequency.

It can only be written when CLMAn is disabled (CLMAnCTL0.CLMAnCLME = 0).

For details, see **Section 13.4.1 (1) Calculation method of the thresholds CLMAnCMPL.CLMAnCMPL[11:0] and CLMAnCMPH.CLMAnCMPH[11:0]**.

**Access:** This register can be read or written in 16-bit units.

**Address:** <CLMAn_base> + $0C_H$

**Value after reset:** $03FF_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | CLMAnCMPH[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 13.8 CLMAnCMPH Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 12 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 11 to 0 | CLMAnCMPH [11:0] | Specifies the upper threshold.<br>• The recommended value is $f_{CLMATMON\ (max)}$ / $f_{CLMATSMP\ (min)}$ × 16 +1.<br>• The minimum value is CLMAnCMPL + $0003_H$. |

### 13.5.4 CLMAnCMPL — CLMAn Compare Register L

This register specifies the lower limit of frequency.
It can only be written when CLMAn is disabled (CLMAnCTL0.CLMAnCLME = 0).

For details, see **Section 13.4.1 (1) Calculation method of the thresholds CLMAnCMPL.CLMAnCMPL[11:0] and CLMAnCMPH.CLMAnCMPH[11:0]**.

**Access:** This register can be read or written in 16-bit units.

**Address:** <CLMAn_base> + $08_H$

**Value after reset:** $0001_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | CLMAnCMPL[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 13.9 CLMAnCMPL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 12 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 11 to 0 | CLMAnCMPL [11:0] | Specifies the lower threshold.<br>• The recommended value is $f_{CLMATMON\ (min)}$ / $f_{CLMATSMP\ (max)}$ × 16 – 1<br>• The minimum value is $0001_H$. |

### 13.5.5 CLMATEST — CLMA Test Register

This register is used to control self-test of CLMA0, CLMA1, and CLMA2.

The correct write sequence using the PROTCMDCLMA register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 C100$_H$

**Value after reset:** 0000 0000$_H$ This register is initialized by a power-up reset PURES (power-on-clear or debugger reset).

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | CLMA2 TESEN | CLMA1 TESEN | CLMA0 TESEN | ERR MSK | MONCL KMSK | RES CLM |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 13.10 CLMATEST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 | CLMA2TESEN | CLMA2 Self-Test Enable/Disable<br>0: Test disabled<br>1: Test enabled |
| 4 | CLMA1TESEN | CLMA1 Self-Test Enable/Disable<br>0: Test disabled<br>1: Test enabled |
| 3 | CLMA0TESEN | CLMA0 Self-Test Enable/Disable<br>0: Test disabled<br>1: Test enabled |
| 2 | ERRMSK | CLMA Test Error Mask Setting<br>Masks a reset request to the reset controller when CLMAn detects an error. When the ERRMSK is set for CLMAn, that CLMAn does not issue a reset request to the reset controller even if it detects an error. The ERRMSK setting is valid only for the CLMAn for which CLMAnTESEN bit is set to 1.<br>0: Reset request signal generation enabled<br>1: Reset request signal generation disabled (masked) |
| 1 | MONCLKMSK | Monitor Clock Mask Setting<br>Fixes the clock input to the CLMAn to low level. The MONCLKMSK setting is valid only for the CLMAn for which CLMAnTESEN bit is set to 1.<br>0: Monitor clock enabled<br>1: Monitor clock disabled (masked) |
| 0 | RESCLM | CLMAn Test Reset Signal Control<br>Initializes CLMAn forcibly. The RESCLM setting is valid only for the CLMAn for which CLMAnTESEN bit is set to 1.<br>0: Reset released<br>1: Reset executed |

### 13.5.6 CLMATESTS — CLMA Test Status Register

This register is used to confirm the self-test result of CLMA0, CLMA1, and CLMA2.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFF8 C104$_H$

**Value after reset:** 0000 0000$_H$ This register is initialized by a power-up reset PURES (power-on-clear or debugger reset).

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CLMA2 ERRS | CLMA1 ERRS | CLMA0 ERRS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 13.11 CLMATESTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 | CLMA2ERRS | CLMA2 Error Status<br>0: Errors are not detected<br>1: Errors are detected |
| 1 | CLMA1ERRS | CLMA1 Error Status<br>0: Errors are not detected<br>1: Errors are detected |
| 0 | CLMA0ERRS | CLMA0 Error Status<br>0: Errors are not detected<br>1: Errors are detected |

### 13.5.7 CLMAnEMU0 — CLMAn Emulation Register 0

This register is used to set pseudo flags for emulation.

This register can be accessed only during break by debugger and is reset by break release.

**Access:** This register can be read or written in 8-bit units.

**Address:** <CLMAn_base> + $18_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CLMAnSLFST | CLMAnSLSLW |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 13.12 CLMAnEMU0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | CLMAnSLFST | Specifies whether $f_{CLMATMON}$ is assumed to be high.<br>0: CLMATMON is assumed to be within the normal frequency range<br>1: CLMATMON is assumed to exceed the upper threshold. |
| 0 | CLMAnSLSLW | Specifies whether $f_{CLMATMON}$ is assumed to be low.<br>0: CLMATMON is assumed to be within the normal frequency range<br>1: CLMATMON is assumed to fall below the lower threshold. |

**CAUTION**

It is prohibited to emulate a low and high CLMATMON at the same time. Thus CLMAnEMU0 must not be set to $03_H$.

## 13.6 Usage Notes for CLMAn

Do not use a clock for which the CLMAn bit indicates an abnormality. The behavior and performance are not guaranteed if such a clock is used. When CLMA0 detects clock abnormality, modifying the clock domain settings is prohibited.

**Table 13.13** provides usage notes for each CLMAn.

**Table 13.13 Usage Notes for CLMAn**

| Monitor Clock | CPU Clock after CLMA Reset Release | Note |
|---|---|---|
| HS IntOSC (CLMA0) | EMCLK*1 | Set ROSCE.ROSCDISTRG to 1.*1<br>Do not set the control registers of the MainOSC and the PLL. After $\overline{\text{CLMA0RES}}$ occurs, modifying any of the clock domain settings is prohibited. |
| MainOSC (CLMA1) | EMCLK (= HS IntOSC) | Do not set control registers of the MainOSC and the PLL. After occurrence of the $\overline{\text{CLMA1RES}}$, do not select a clock whose source clock is the MainOSC or the PLL. |
| PLL (CLMA2) | EMCLK (= HS IntOSC) | Do not set control registers of the PLL. After occurrence of the $\overline{\text{CLMA2RES}}$, do not select a clock whose source clock is the PLL. |

Note 1. The state of EMCLK after reset by $\overline{\text{CLMA0RES}}$ depends on the state of HS IntOSC oscillation. If HS IntOSC is completely stopped, LS IntOSC is supplied as EMCLK. If HS IntOSC continues oscillating but the frequency is outside of the lower threshold and the upper threshold of CLMA0, HS IntOSC is still supplied as EMCLK. After setting the ROSCE.ROSCDISTRG bit to 1, EMCLK is switched from HS IntOSC to LS IntOSC.

# Section 14 Stand-By Controller (STBC)

This section describes the functions of the stand-by controller (STBC), the registers, and various stand-by modes.

## 14.1 Functions

### 14.1.1 Types of Stand-By Mode

The RH850/F1K supports STOP mode and DeepSTOP mode for system-level low power status. In addition, the RH850/F1K supports cyclic operation (Cyclic RUN mode and Cyclic STOP mode) which supports low-power operation of limited functions. Transition between each mode is described in the **Section 14.1.5, Transition to Stand-By Mode**.

- RUN mode
  RUN mode is a normal operation mode where the CPU is operating and all of other modules can operate. The CPU can enter “HALT” state by executing the “HALT” instruction to stop its operation in this mode.
- STOP mode
  STOP mode is a chip-level stand-by mode in which the clock supply to a certain clock domain can be stopped.
  STOP mode is entered when the STBC0STPT.STBC0STPTRG bit is set to 1.
  The clock supply to clock domains can continue even in STOP mode by setting CKSC_xxx_STPM.xxxSTPMSK = 1. For details on the CKSC_xxx_STPM register, see **Section 12, Clock Controller**.
- DeepSTOP mode
  DeepSTOP mode is a chip-level stand-by mode to reduce power consumption further than STOP mode. In addition to the clock supply stop, the power supply to the Isolated area is switched off.
  DeepSTOP mode is entered when the STBC0PSC.STBC0DISTRG is set to 1.
- Cyclic RUN mode
  Cyclic RUN mode is a low-power operation mode in which limited modules can operate at low speed. In this mode, only the CPU, the peripheral functions in the Always-On area, RLIN3 and CSIG0 can operate. Data/Code flash memory and PLL are not available.
  The CPU executes the instructions in the retention RAM.
  In this mode, the CPU can issue the “HALT” instruction to enter HALT state.
  The mode transition to Cyclic RUN mode from DeepSTOP mode is triggered by wake-up factor 2, and the mode transition to Cyclic RUN mode from Cyclic STOP mode is triggered by either wake-up factor 1 or wake-up factor 2.
  CSIG0 cannot be wake-up factor.
- Cyclic STOP mode
  Cyclic STOP mode is a STOP mode in cyclic operation, and the CPU halts its operation.
  This mode is entered by setting the STBC0STPT.STBC0STPTRG bit to 1 in Cyclic RUN mode.

## 14.1.2 Wake-Up Control

### 14.1.2.1 Wake-Up Factors for Stand-By Modes

The stand-by controller can initiate return from stand-by mode by the following wake-up factors.

**Table 14.1 Wake-Up Factor 1 (1/2)**

| Wake-Up Factor | Unit | STOP →RUN | DeepSTOP →RUN | Cyclic RUN →RUN*1 | Cyclic STOP →RUN*1 |
|---|---|---|---|---|---|
| TNMI | Port | √ | √ | √ | √ |
| WDTA0NMI | WDTA0 | √ | √ | √ | √ |
| INTLVIL*3 | LVI | √ | √ | √ | √ |
| INTP0 | Port | √ | √ | √ | √ |
| INTP1 | Port | √ | √ | √ | √ |
| INTP2 | Port | √ | √ | √ | √ |
| INTWDTA0 | WDTA0 | √ | √ | √ | √ |
| INTP3 | Port | √ | √ | √ | √ |
| INTP4 | Port | √ | √ | √ | √ |
| INTP5 | Port | √ | √ | √ | √ |
| INTP10 | Port | √ | √ | √ | √ |
| INTP11 | Port | √ | √ | √ | √ |
| WUTRG1 | LPS0 | √ | √ | √ | √ |
| INTTAUJ0I0 | TAUJ0 | √ | √ | √ | √ |
| INTTAUJ0I1 | TAUJ0 | √ | √ | √ | √ |
| INTTAUJ0I2 | TAUJ0 | √ | √ | √ | √ |
| INTTAUJ0I3 | TAUJ0 | √ | √ | √ | √ |
| WUTRG0 | LPS0 | √ | √ | √ | √ |
| INTP6 | Port | √ | √ | √ | √ |
| INTP7 | Port | √ | √ | √ | √ |
| INTP8 | Port | √ | √ | √ | √ |
| INTP12 | Port | √ | √ | √ | √ |
| INTP9 | Port | √ | √ | √ | √ |
| INTP13 | Port | √ | √ | √ | √ |
| INTP14 | Port | √ | √ | √ | √ |
| INTP15 | Port | √ | √ | √ | √ |
| INTRTCA01S | RTCA0 | √ | √ | √ | √ |
| INTRTCA0AL | RTCA0 | √ | √ | √ | √ |
| INTRTCA0R | RTCA0 | √ | √ | √ | √ |
| INTDCUTDI | JTAG | √ | √ | √ | √ |
| INTKR0 | KR0 | √ | — | — | — |
| INTRCANGRECC0*2 | RSCAN0 | √ | — | — | — |
| INTRCAN0REC*2 | RSCAN0 | √ | — | — | — |
| INTRCAN1REC*2 | RSCAN0 | √ | — | — | — |
| INTRCAN2REC*2 | RSCAN0 | √ | — | — | — |
| INTRCAN3REC*2 | RSCAN0 | √ | — | — | — |
| INTRCAN4REC*2 | RSCAN0 | √ | — | — | — |

**Table 14.1 Wake-Up Factor 1 (2/2)**

| Wake-Up Factor | Unit | STOP →RUN | DeepSTOP →RUN | Cyclic RUN →RUN*1 | Cyclic STOP →RUN*1 |
|---|---|---|---|---|---|
| INTRCAN5REC*2 | RSCAN0 | √ | — | — | — |
| INTRCANGRECC1*2 | RSCAN1 | √ | — | — | — |
| INTRCAN6REC*2 | RSCAN1 | √ | — | — | — |

Note 1. The mode returns to RUN mode via DeepSTOP mode. When the transition from Cyclic STOP to Cyclic RUN is made by wake-up factor 1, if the transition to DeepSTOP by STBC0PSC.STBC0DISTRG is made without clearing wake-up factor 1, the transition to RUN mode is made.

Note 2. By using the INTP external interrupt assigned to the alternate-function pin shared with the CAN reception pin, wake-up from stand-by modes such as DeepSTOP is possible. As the trigger for waking up from DeepSTOP, use a pin of port P0 or P1, which is assigned to the Always-On area.
To clear the wake-up factor flag, the interrupt request must be cleared in each CAN module.

Note 3. Cannot be cleared while REGVCC is below the reference voltage (REGVCC (min.) < VLVI).
To clear the wake-up factor flag (WUF0[2]), WUFC0[2] must be set while WUFMSK0[2] = 1 and REGVCC is above the reference voltage (REGVCC (min.) > VLVI). The INTLVIH interrupt can be used to check that REGVCC is above the reference voltage.

**Table 14.2 Wake-Up Factor 2**

| Wake-Up Factor | Unit | DeepSTOP → Cyclic RUN | Cyclic STOP → Cyclic RUN |
|---|---|---|---|
| INTADCA0I0 | ADCA0 | √*1 | √*1 |
| INTADCA0I1 | ADCA0 | √*1 | √*1 |
| INTADCA0I2 | ADCA0 | √*1 | √*1 |
| INTRLIN30 | RLIN30 | — | √ |
| INTTAUJ0I0 | TAUJ0 | √ | √ |
| INTTAUJ0I1 | TAUJ0 | √ | √ |
| INTTAUJ0I2 | TAUJ0 | √ | √ |
| INTTAUJ0I3 | TAUJ0 | √ | √ |
| INTRLIN31 | RLIN31 | — | √ |
| INTRLIN32 | RLIN32 | — | √ |
| INTRTCA01S | RTCA0 | √ | √ |
| INTRTCA0AL | RTCA0 | √ | √ |
| INTRTCA0R | RTCA0 | √ | √ |
| INTRLIN33 | RLIN33 | — | √ |
| INTRLIN34 | RLIN34 | — | √ |
| INTRLIN35 | RLIN35 | — | √ |

Note 1. These wake-up factors are only available in LPS analog input mode.

**CAUTION**

**For the pins of the function used for the wake-up factors from DeepSTOP, use the multiplexed functions of the ports assigned to the Always-On area.**

#### 14.1.2.2 Setting of Wake-Up Factors

Wake-up factors for returning from stand-by modes are controlled by the following stand-by controller registers:

- Wake-up factor registers: WUF0, WUF20, WUF_ISO0

  Upon occurrence of an effective wake-up factor, the associated wake-up factor flag is set to 1. By checking these registers and their flags, it is possible to identify the wake-up factor.

- Wake-up factor mask registers: WUFMSK0, WUFMSK20, WUFMSK_ISO0

  Each bit of these registers is assigned to a certain wake-up factor. Wake-up by this factor is enabled if its mask bit is set to 0. Wake-up factors assigned to both wake-up factor 1 and 2 should not be enabled at the same time.

- Wake-up factor clear registers: WUFC0, WUFC20, WUFC_ISO0

  By setting the applicable bits in these registers to 1, the wake-up factor bit (WUFy) in the wake-up factor registers (WUF0, WUF20, WUF_ISO0) can be cleared.

**NOTE**

The wake-up factor flags in the wake-up factor registers (WUF0, WUF20, and WUF_ISO0) only indicate the occurrence of wake-up factor. These flags do not indicate a transition from stand-by mode to normal mode.

The assignment of the wake-up factors to the control register bits and status register bits are shown in the following tables.

For details about the wake-up control and status registers, see **Section 14.2.2.3, WUF0/WUF20/WUF_ISO0 — Wake-Up Factor Registers**, **Section 14.2.2.4, WUFMSK0/WUFMSK20/WUFMSK_ISO0 — Wake-Up Factor Mask Registers**, and **Section 14.2.2.5, WUFC0/WUFC20/WUFC_ISO0 — Wake-Up Factor Clear Registers**.

Table 14.3 Wake-Up Factor 1 Register Assignment

| Wake-Up Factor | Register Bit Assignment | | | Unit | 100 pins | 144 pins | 176 pins |
|---|---|---|---|---|---|---|---|
| TNMI | WUF0[0] | WUFMSK0[0] | WUFC0[0] | Port | √ | √ | √ |
| WDTA0NMI | WUF0[1] | WUFMSK0[1] | WUFC0[1] | WDTA0 | √ | √ | √ |
| INTLVIL | WUF0[2] | WUFMSK0[2] | WUFC0[2] | LVI | √ | √ | √ |
| INTP0 | WUF0[5] | WUFMSK0[5] | WUFC0[5] | Port | √ | √ | √ |
| INTP1 | WUF0[6] | WUFMSK0[6] | WUFC0[6] | Port | √ | √ | √ |
| INTP2 | WUF0[7] | WUFMSK0[7] | WUFC0[7] | Port | √ | √ | √ |
| INTWDTA0 | WUF0[8] | WUFMSK0[8] | WUFC0[8] | WDTA0 | √ | √ | √ |
| INTP3 | WUF0[9] | WUFMSK0[9] | WUFC0[9] | Port | √ | √ | √ |
| INTP4 | WUF0[10] | WUFMSK0[10] | WUFC0[10] | Port | √ | √ | √ |
| INTP5 | WUF0[11] | WUFMSK0[11] | WUFC0[11] | Port | √ | √ | √ |
| INTP10 | WUF0[12] | WUFMSK0[12] | WUFC0[12] | Port | √ | √ | √ |
| INTP11 | WUF0[13] | WUFMSK0[13] | WUFC0[13] | Port | √ | √ | √ |
| WUTRG1 | WUF0[14] | WUFMSK0[14] | WUFC0[14] | LPS0 | √ | √ | √ |
| INTTAUJ0I0 | WUF0[15] | WUFMSK0[15] | WUFC0[15] | TAUJ0 | √ | √ | √ |
| INTTAUJ0I1 | WUF0[16] | WUFMSK0[16] | WUFC0[16] | TAUJ0 | √ | √ | √ |
| INTTAUJ0I2 | WUF0[17] | WUFMSK0[17] | WUFC0[17] | TAUJ0 | √ | √ | √ |
| INTTAUJ0I3 | WUF0[18] | WUFMSK0[18] | WUFC0[18] | TAUJ0 | √ | √ | √ |
| WUTRG0 | WUF0[19] | WUFMSK0[19] | WUFC0[19] | LPS0 | √ | √ | √ |
| INTP6 | WUF0[20] | WUFMSK0[20] | WUFC0[20] | Port | √ | √ | √ |
| INTP7 | WUF0[21] | WUFMSK0[21] | WUFC0[21] | Port | √ | √ | √ |
| INTP8 | WUF0[22] | WUFMSK0[22] | WUFC0[22] | Port | √ | √ | √ |
| INTP12 | WUF0[23] | WUFMSK0[23] | WUFC0[23] | Port | √ | √ | √ |
| INTP9 | WUF0[24] | WUFMSK0[24] | WUFC0[24] | Port | — | √ | √ |
| INTP13 | WUF0[25] | WUFMSK0[25] | WUFC0[25] | Port | √ | √ | √ |
| INTP14 | WUF0[26] | WUFMSK0[26] | WUFC0[26] | Port | — | √ | √ |
| INTP15 | WUF0[27] | WUFMSK0[27] | WUFC0[27] | Port | — | √ | √ |
| INTRTCA01S | WUF0[28] | WUFMSK0[28] | WUFC0[28] | RTCA0 | — | √ | √ |
| INTRTCA0AL | WUF0[29] | WUFMSK0[29] | WUFC0[29] | RTCA0 | — | √ | √ |
| INTRTCA0R | WUF0[30] | WUFMSK0[30] | WUFC0[30] | RTCA0 | — | √ | √ |
| INTDCUTDI | WUF0[31] | WUFMSK0[31] | WUFC0[31] | JTAG | √ | √ | √ |
| INTKR0 | WUF_ISO0[1] | WUFMSK_ISO0[1] | WUFC_ISO0[1] | KR0 | √ | √ | √ |
| INTRCANGRECC0 | WUF_ISO0[2] | WUFMSK_ISO0[2] | WUFC_ISO0[2] | RSCAN0 | √ | √ | √ |
| INTRCAN0REC | WUF_ISO0[3] | WUFMSK_ISO0[3] | WUFC_ISO0[3] | RSCAN0 | √ | √ | √ |
| INTRCAN1REC | WUF_ISO0[4] | WUFMSK_ISO0[4] | WUFC_ISO0[4] | RSCAN0 | √ | √ | √ |
| INTRCAN2REC | WUF_ISO0[5] | WUFMSK_ISO0[5] | WUFC_ISO0[5] | RSCAN0 | √ | √ | √ |
| INTRCAN3REC | WUF_ISO0[6] | WUFMSK_ISO0[6] | WUFC_ISO0[6] | RSCAN0 | √ | √ | √ |
| INTRCAN4REC | WUF_ISO0[7] | WUFMSK_ISO0[7] | WUFC_ISO0[7] | RSCAN0 | √ | √ | √ |
| INTRCAN5REC | WUF_ISO0[8] | WUFMSK_ISO0[8] | WUFC_ISO0[8] | RSCAN0 | √ | √ | √ |
| INTRCANGRECC1 | WUF_ISO0[9] | WUFMSK_ISO0[9] | WUFC_ISO0[9] | RSCAN1 | — | — | √ |
| INTRCAN6REC | WUF_ISO0[10] | WUFMSK_ISO0[10] | WUFC_ISO0[10] | RSCAN1 | — | — | √ |

Table 14.4 Wake-Up Factor 2 Register Assignment

| Wake-Up Factor | Register Bit Assignment | | | Unit | 100 pins | 144 pins | 176 pins |
|---|---|---|---|---|---|---|---|
| INTADCA0I0 | WUF20[0] | WUFMSK20[0] | WUFC20[0] | ADCA0 | √ | √ | √ |
| INTADCA0I1 | WUF20[1] | WUFMSK20[1] | WUFC20[1] | ADCA0 | √ | √ | √ |
| INTADCA0I2 | WUF20[2] | WUFMSK20[2] | WUFC20[2] | ADCA0 | √ | √ | √ |
| INTRLIN30 | WUF20[3] | WUFMSK20[3] | WUFC20[3] | RLIN30 | √ | √ | √ |
| INTTAUJ0I0 | WUF20[4] | WUFMSK20[4] | WUFC20[4] | TAUJ0 | √ | √ | √ |
| INTTAUJ0I1 | WUF20[5] | WUFMSK20[5] | WUFC20[5] | TAUJ0 | √ | √ | √ |
| INTTAUJ0I2 | WUF20[6] | WUFMSK20[6] | WUFC20[6] | TAUJ0 | √ | √ | √ |
| INTTAUJ0I3 | WUF20[7] | WUFMSK20[7] | WUFC20[7] | TAUJ0 | √ | √ | √ |
| INTRLIN31 | WUF20[8] | WUFMSK20[8] | WUFC20[8] | RLIN31 | √ | √ | √ |
| INTRLIN32 | WUF20[9] | WUFMSK20[9] | WUFC20[9] | RLIN32 | √ | √ | √ |
| INTRTCA01S | WUF20[10] | WUFMSK20[10] | WUFC20[10] | RTCA0 | — | √ | √ |
| INTRTCA0AL | WUF20[11] | WUFMSK20[11] | WUFC20[11] | RTCA0 | — | √ | √ |
| INTRTCA0R | WUF20[12] | WUFMSK20[12] | WUFC20[12] | RTCA0 | — | √ | √ |
| INTRLIN33 | WUF20[13] | WUFMSK20[13] | WUFC20[13] | RLIN33 | √ | √ | √ |
| INTRLIN34 | WUF20[14] | WUFMSK20[14] | WUFC20[14] | RLIN34 | — | √ | √ |
| INTRLIN35 | WUF20[15] | WUFMSK20[15] | WUFC20[15] | RLIN35 | — | √ | √ |

### 14.1.3 On-Chip Debug Wake-Up

The On-Chip Debug unit (OCD) generates a wake-up event while the microcontroller runs the application program in the following cases:

- The debugger issues a stop request
- A breakpoint is hit

In either case all stand-by modes are terminated, provided the OCD debug event is enabled as a wake-up factor via the WUFMSK0 register.

**CAUTION**

**If the OCD wake-up event is disabled, it is not possible to wake up from stand-by modes via an On-chip debugger request.**

**The OCD wake-up event can be enabled as a wake-up factor for all stand-by modes by setting WUFMSK0[31] = 0.**

**When the hot plug-in function is used, make sure to enable the OCD wake-up event and return from stand-by mode by INTDCUTDI interrupt.**

### 14.1.4 I/O Buffer Control

This section describes the behavior of the I/O buffers during various stand-by modes.

The port groups in the Isolated area support the I/O buffer hold state. The port groups in the Always-On area remain their state before entering stand-by mode.
For details on the port group assignment to the Isolated area and to the Always-On area, see **Section 10, Power Supply Circuit**.

#### 14.1.4.1 I/O Buffer Hold State

During the I/O buffer hold state, the I/O buffers maintains the state it was in before entering this state. Therefore, no external or internal signal can change the state of the I/O buffer until the I/O buffer hold state is terminated.

#### 14.1.4.2 I/O Buffers during STOP Mode

In STOP mode, the I/O buffers remain their state before entering STOP mode (I/O buffer hold state is not entered).

#### 14.1.4.3 I/O Buffers during DeepSTOP Mode

In DeepSTOP mode, the I/O buffers of port groups in the Isolated area transition to I/O buffer hold state.

After wake-up from DeepSTOP, the I/O buffers remain in I/O buffer hold state until the state is canceled by software. To cancel I/O buffer hold state, follow the steps shown below.

1. Re-configure the peripheral or port function.
2. Set IOHOLD.IOHOLD = 0.

The following table is a summary of the I/O buffer in the Isolated area during stand-by mode and after wake-up.

**Table 14.5 Buffer Operation during Stand-by Mode and after Wake-Up (I/O buffers in the Isolated area)**

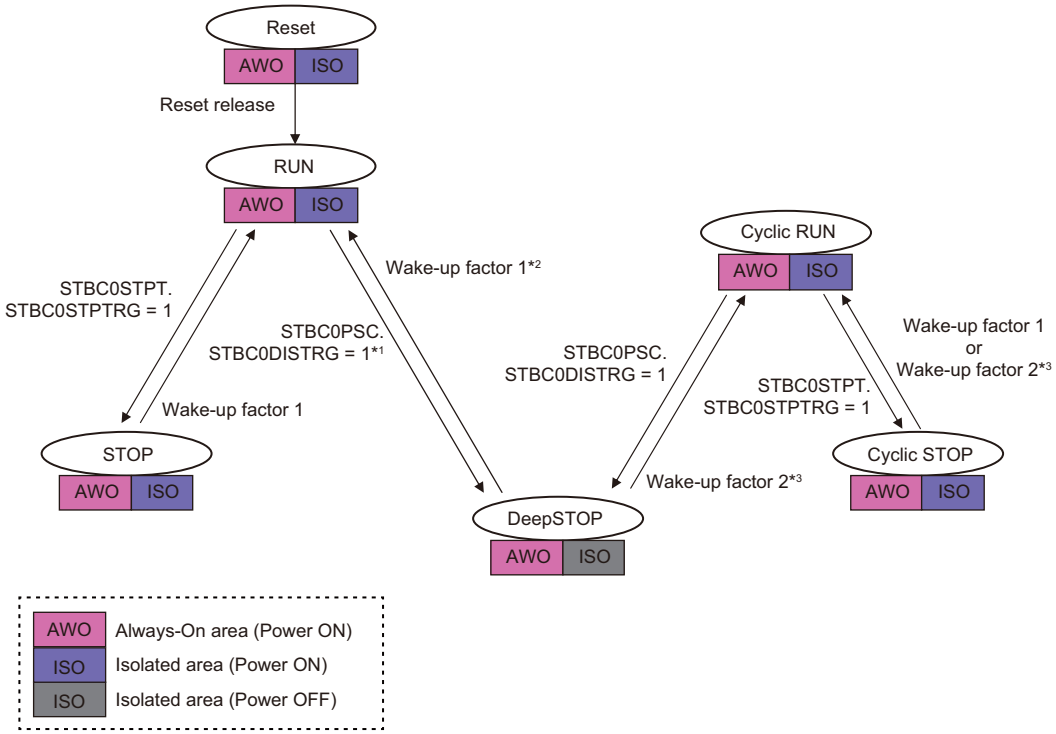
| | Before Stand-By | During Stand-By | After Wake-Up |
|---|---|---|---|
| STOP mode | Normal operation | | |
| DeepSTOP mode | Normal operation | I/O buffer hold state | I/O buffer hold state *1 |

Note 1. Set the IOHOLD.IOHOLD bit to “0” to release the I/O buffer hold state.

The port groups in the Always-On area don't support I/O buffer hold state. They continue operation and remain its state before entering DeepSTOP. In the case an alternative function of modules in Isolated area is assigned to the pin in the Always-On area, the state of the I/O buffer may change in the transition to the DeepSTOP due to initialization of the modules in the Isolated area by ISORES. To avoid this behavior, it is recommended to change to function of modules in Always-On area (e. g. Port mode) before entering DeepSTOP.

### 14.1.5 Transition to Stand-By Mode

The figure below shows transition between RUN mode and stand-by mode.




Note 1. Set a clock other than the PLL clock as the CPU clock source.
Note 2. To return from DeepSTOP to RUN, ISO peripheral function registers must be set again. The CPU starts the operation from the reset address on the code flash.
Note 3. The CPU starts the operation from the reset address on the retention RAM.

**NOTE**

The mode transition time can be optimized by setting 1 to AWDTADSTPMSK bit of CKSC_AWDTAD_STPM register.

**Figure 14.1 Transition to Stand-By Mode**

### 14.1.6 Clock Supply

The clock supply to the stand-by controller is shown in the following table.

**Table 14.6 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name |
|---|---|---|
| Stand-by controller | Register access clock | CPUCLK2 |
| | | EMCLK |

## 14.2 Registers

### 14.2.1 List of Registers

The following table lists the stand-by controller registers.

**Table 14.7 List of Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| STBC0 | Power save control register | STBC0PSC | FFF8 $0100_H$ |
| | Power stop trigger register | STBC0STPT | FFF8 $0110_H$ |
| STBC_WUF0 | Wake-up factor registers | WUF0 | FFF8 $0400_H$ |
| STBC_WUF20 | | WUF20 | FFF8 $0520_H$ |
| STBC_WUFISO | | WUF_ISO0 | FFF8 $8110_H$ |
| STBC_WUF0 | Wake-up factor mask registers | WUFMSK0 | FFF8 $0404_H$ |
| STBC_WUF20 | | WUFMSK20 | FFF8 $0524_H$ |
| STBC_WUFISO | | WUFMSK_ISO0 | FFF8 $8114_H$ |
| STBC_WUF0 | Wake-up factor clear registers | WUFC0 | FFF8 $0408_H$ |
| STBC_WUF20 | | WUFC20 | FFF8 $0528_H$ |
| STBC_WUFISO | | WUFC_ISO0 | FFF8 $8118_H$ |
| STBC_IOHOLD | I/O buffer hold control register | IOHOLD | FFF8 $0B00_H$ |

## 14.2.2 Details of Stand-By Controller Control Registers

### 14.2.2.1 STBC0PSC — Power Save Control Register

The correct write sequence using the PROTCMD0 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 $0100_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | STBC0DISTRG | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R |

**Table 14.8 STBC0PSC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 1 | STBC0DISTRG | 0: No effect<br>1: Transition to DeepSTOP mode<br>This bit is cleared automatically after transition to the DeepSTOP mode. |
| 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

### 14.2.2.2 STBC0STPT — Power Stop Trigger Register

The correct write sequence using the PROTCMD0 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 $0110_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | STBC0STPTRG |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 14.9 STBC0STPT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | STBC0STPTRG | 0: No effect.<br>1: Transition to STOP mode<br>– In RUN mode: Transition to STOP mode<br>– In Cyclic RUN mode: Transition to Cyclic STOP mode<br>This bit is cleared automatically after transition to the STOP / Cyclic STOP mode. |

### 14.2.2.3 WUF0/WUF20/WUF_ISO0 — Wake-Up Factor Registers

These registers indicate the generation of wake-up factors.

WUF0 and WUF20 are initialized by all reset sources except the transition to DeepSTOP mode (AWORES). WUF_ISO0 is initialized by all reset sources (ISORES).

**Access:** These registers are read-only registers that can be read in 32-bit units.

**Address:** WUF0: FFF8 $0400_H$
WUF20: FFF8 $0520_H$
WUF_ISO0: FFF8 $8110_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WUF31 | WUF30 | WUF29 | WUF28 | WUF27 | WUF26 | WUF25 | WUF24 | WUF23 | WUF22 | WUF21 | WUF20 | WUF19 | WUF18 | WUF17 | WUF16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WUF15 | WUF14 | WUF13 | WUF12 | WUF11 | WUF10 | WUF09 | WUF08 | WUF07 | WUF06 | WUF05 | WUF04 | WUF03 | WUF02 | WUF01 | WUF00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 14.10 WUF0/WUF20/WUF_ISO0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | WUFy | Indicates the generation of a wake-up factor.<br>0: Wake-up factor is not generated<br>1: Wake-up factor is generated |

**NOTE**

While the WUFMSKy bit in the wake-up factor mask register is 1, WUFy is not set to 1 at the generation of a wake-up factor.

#### Wake-Up Factors

As for the assignment of wake-up factors to the wake-up factor register bits, see **Table 14.3, Wake-Up Factor 1 Register Assignment** and **Table 14.4, Wake-Up Factor 2 Register Assignment**.

The bit to which a wake-up factor is not assigned is read as the value “0”.

### 14.2.2.4 WUFMSK0/WUFMSK20/WUFMSK_ISO0 — Wake-Up Factor Mask Registers

These registers enable wake-up factors.

WUFMSK0 and WUFMSK20 are initialized by all reset sources except the transition to DeepSTOP mode (AWORES). WUFMSK_ISO0 is initialized by all reset sources (ISORES).

**Access:** These registers can be read or written in 32-bit units.

**Address:** WUFMSK0: FFF8 0404$_H$
WUFMSK20: FFF8 0524$_H$
WUFMSK_ISO0: FFF8 8114$_H$

**Value after reset:** WUFMSK0: FFFF FFFF$_H$
WUFMSK20: FFFF FFFF$_H$
WUFMSK_ISO0: FFFF FFFF$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WUFMSK31 | WUFMSK30 | WUFMSK29 | WUFMSK28 | WUFMSK27 | WUFMSK26 | WUFMSK25 | WUFMSK24 | WUFMSK23 | WUFMSK22 | WUFMSK21 | WUFMSK20 | WUFMSK19 | WUFMSK18 | WUFMSK17 | WUFMSK16 |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WUFMSK15 | WUFMSK14 | WUFMSK13 | WUFMSK12 | WUFMSK11 | WUFMSK10 | WUFMSK09 | WUFMSK08 | WUFMSK07 | WUFMSK06 | WUFMSK05 | WUFMSK04 | WUFMSK03 | WUFMSK02 | WUFMSK01 | WUFMSK00 |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 14.11 WUFMSK0/WUFMSK20/WUFMSK_ISO0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | WUFMSKy | Enables/disables a wake-up factor.<br>0: Wake-up factor is enabled<br>1: Wake-up factor is disabled |

**NOTE**

While the WUFMSKy bit is 1, WUFy of the wake-up factor register is not set to 1 at the generation of a wake-up factor.

#### Wake-Up Factors

As for the assignment of wake-up factors to the wake-up factor register bits, see **Table 14.3, Wake-Up Factor 1 Register Assignment** and **Table 14.4, Wake-Up Factor 2 Register Assignment**.

When writing to these registers, write the value “1” to the bits to which wake-up factors are not assigned.

### 14.2.2.5 WUFC0/WUFC20/WUFC_ISO0 — Wake-Up Factor Clear Registers

These registers clear the WUFy bits in the wake-up factor registers.

**Access:** These registers are write-only registers that can be written in 32-bit units.

**Address:** WUFC0: FFF8 $0408_H$
WUFC20: FFF8 $0528_H$
WUFC_ISO0: FFF8 $8118_H$

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WUFC 31 | WUFC 30 | WUFC 29 | WUFC 28 | WUFC 27 | WUFC 26 | WUFC 25 | WUFC 24 | WUFC 23 | WUFC 22 | WUFC 21 | WUFC 20 | WUFC 19 | WUFC 18 | WUFC 17 | WUFC 16 |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WUFC 15 | WUFC 14 | WUFC 13 | WUFC 12 | WUFC 11 | WUFC 10 | WUFC 09 | WUFC 08 | WUFC 07 | WUFC 06 | WUFC 05 | WUFC 04 | WUFC 03 | WUFC 02 | WUFC 01 | WUFC 00 |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 14.12 WUFC0/WUFC20/WUFC_ISO0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | WUFCy | Clears the wake-up factor bit WUFy in the wake-up factor registers.<br>0: WUFy is not modified<br>1: WUFy is cleared |

#### Wake-Up Factors

As for the assignment of wake-up factors to the wake-up factor register bits, see **Table 14.3, Wake-Up Factor 1 Register Assignment** and **Table 14.4, Wake-Up Factor 2 Register Assignment**.

When writing to these registers, write the value “0” to the bits to which wake-up factors are not assigned.

### 14.2.2.6 IOHOLD — I/O Buffer Hold Control Register

This register specifies the hold state of the I/O buffer in DeepSTOP mode. The correct write sequence using the PROTCMD0 register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

This register is initialized by all reset sources except the transition to DeepSTOP mode (AWORES).

**Access:** This register can be read or written in 32-bit units.

**Address:** FFF8 0B00$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IOHOLD |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 14.13 IOHOLD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | IOHOLD | 0: I/O hold state is released<br>1: I/O hold state<br>This bit is automatically set to 1 at the transition to DeepSTOP mode. Setting this bit to 1 by software is prohibited. To release the I/O hold state after the wake-up, set this bit to 0 by software. |

## 14.3 Mode Transition

This section explains the mode transition procedures.

### 14.3.1 STOP Mode

In STOP mode, most of the clock supplies to the Always-On area and the Isolated area are stopped. The clock stop mask registers control clock supply to related clock domains in stand-by mode. Stop all of the peripheral functions before transition to STOP mode if the clock supply to the function will be stopped in STOP mode.

The transition procedure (example) to STOP mode is shown below.

#### Preparation for stand-by

- Stop all of the peripheral functions to which the clock supply is to be stopped.
- Disable the interrupt handling by issuing the CPU instruction “DI”.
- Set the interrupt control registers.
  - Clear the interrupt flag (ICxxx.RFxxx = 0).
  - Mask the interrupts for non-wake-up factors (ICxxx.MKxxx = 1).
  - Release the masks of the interrupts for wake-up factors (ICxxx.MKxxx = 0).
- Set the wake-up related registers.
  - Clear the wake-up factor flags (the WUFC0/WUFC_ISO0 registers).
  - Mask the non-wake-up factor (the WUFMSK0/WUFMSK_ISO0 registers).
  - Release the masks of the wake-up factors (the WUFMSK0/WUFMSK_ISO0 registers).
- Set the clock stop mask register to select the clock domains to be stopped and the ones to continue operating (using the CKSC_xxx_STPM.xxxxSTPMSK bit).
- Specify whether to oscillate or stop each clock source. In addition, set the clock stop mask register to select the clock sources to be stopped and the ones to continue operating (using the MOSCSTPMSK bit in the MOSCSTPM register and the ROSCSTPMSK bit in the ROSCSTPM register).

#### Start of stand-by

Set the STBC0STPTRG bit in the STBC0STPT register to 1 to transition to STOP mode.

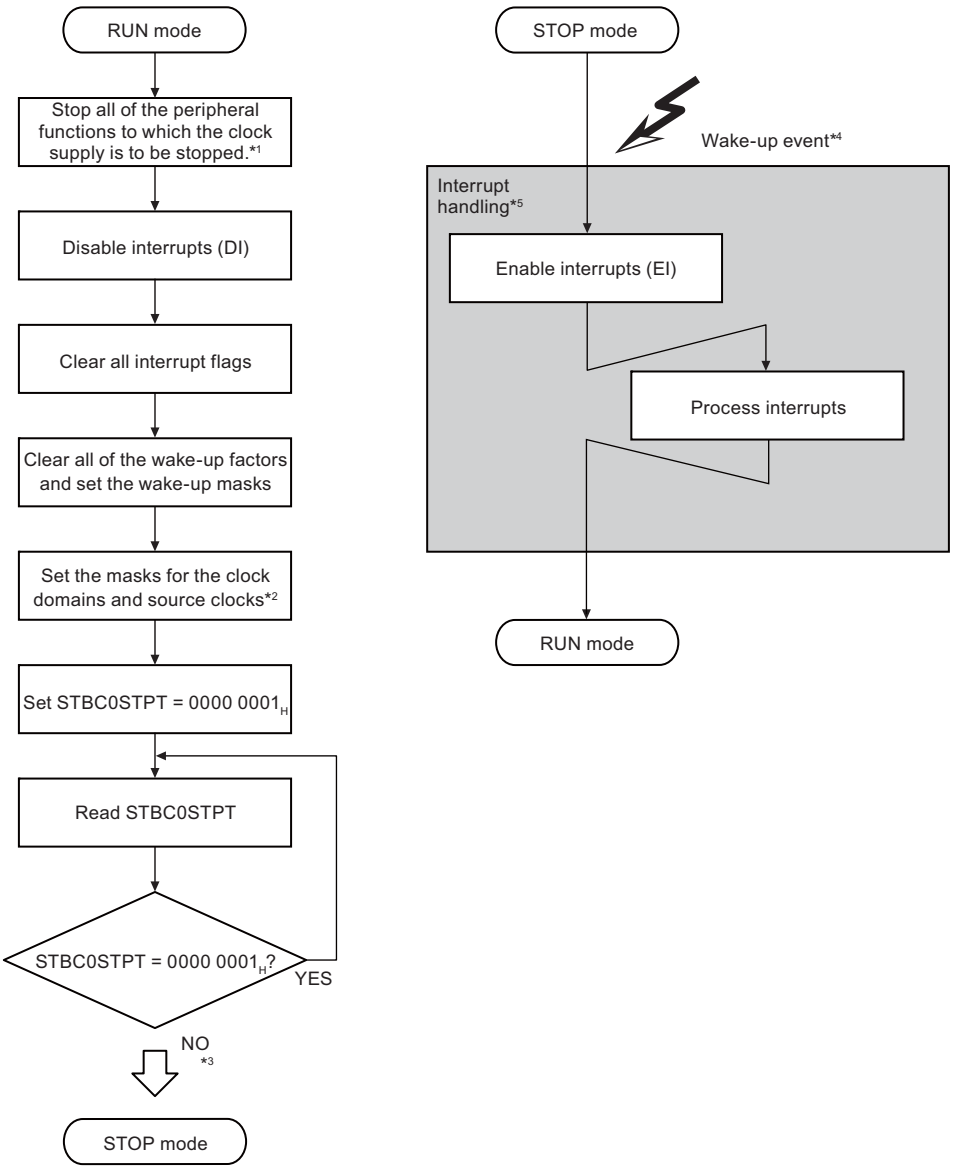
#### End of stand-by

When a wake-up factor is generated, the microcontroller returns from STOP mode.

#### Wake-up handling

The generation of the wake-up factors can be determined by the wake-up factor flags (WUF0, WUF_ISO0).

When an interrupt is enabled by the CPU instruction “EI”, the generated wake-up interrupt will be executed.




Note 1. When the operation of the peripheral function is stopped during operating due to the transition to the STOP mode, the operation of the peripheral function may be incorrect. Before the transition to the STOP mode, stop all of the peripheral functions to which the clock supply is to be stopped.

Note 2. Though the clock mask can be set before this flow is started, it must be set before $0000\ 0001_H$ is written to STBC0STPT.

Note 3. The clock supply to the CPU is stopped and the operation transitions to the STOP mode while checking that STBC0STPT = $0000\ 0001_H$.

Note 4. STBC0STPT is set to $0000\ 0000_H$ at the generation of a wake-up factor. The generated wake-up factor can be checked by the WUF0 and WUF_ISO0 registers.

Note 5. This processing is optional. It is required when executing the interrupt handling after the wake-up.

**Figure 14.2 Example of STOP Mode Transition**

### 14.3.2 DeepSTOP Mode

In DeepSTOP mode, the clock supply to all areas and the power supply to the Isolated area are stopped. However, clock supply to peripheral functions in the Always-On area can be continued by setting the clock stop mask register.

Select the clock other than the PLL as the CPU operating clock, before the transition to DeepSTOP mode.

The transition procedure (example) to DeepSTOP mode is shown below.

#### Preparation for stand-by

- Stop all of the peripheral functions to which the clock supply is to be stopped.
- Disable the interrupt handling by issuing the CPU instruction “DI”.
- Set the interrupt control registers.
  - Clear the interrupt flag (ICxxx.RFxxx = 0).
  - Mask the interrupts for non-wake-up factors (ICxxx.MKxxx = 1).
  - Release the masks of the interrupts for wake-up factors (ICxxx.MKxxx = 0).
- Set the wake-up related registers.
  - Clear the wake-up factor flags (the WUFC0/WUFC20 registers).
  - Mask the non-wake-up factor (the WUFMSK0/WUFMSK20 registers).
  - Release the masks of the wake-up factors (the WUFMSK0/WUFMSK20 registers).

**CAUTION**

**When a wake-up factor is assigned to both wake-up factor 1 registers and wake-up factor 2 registers, it can be used only in one of them.**

- Set the clock stop mask register to select the clock domains to be stopped and the ones to continue operating (using the CKSC_xxx_STPM.xxxxSTPMSK bit).
- Specify whether to oscillate or stop each clock source. In addition, set the clock stop mask register to select the clock sources to be stopped and the ones to continue operating (using the MOSCSTPMSK bit in the MOSCSTPM register and the ROSCSTPMSK bit in the ROSCSTPM register).

#### Start of stand-by

Set the STBC0DISTRG bit in the STBC0PSC register to 1 to transition to DeepSTOP mode.

#### End of stand-by

When a wake-up factor is generated, the microcontroller returns from DeepSTOP mode.

### Wake-up handling

- When returned from DeepSTOP mode due to wake-up factor 1, the microcontroller starts the operation from the reset vector address.

  If one of the following interrupts has been generated before recovery from DeepSTOP mode to RUN mode, the microcontroller restarts operation from the exception handler address:

  – FENMI: FENMI handler address ($E0_H$)
  – FEINT: FEINT handler address ($F0_H$)

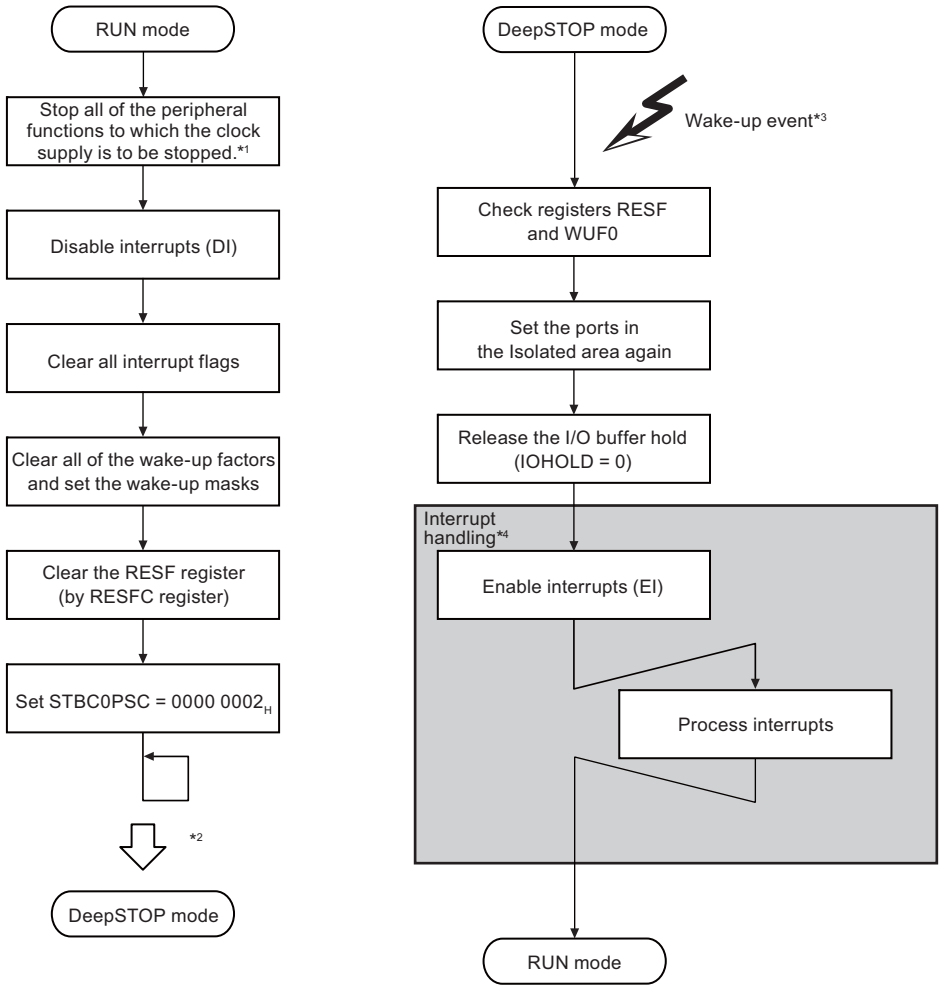
  Note that the General-purpose registers and Local RAM are undefined value after return from DeepSTOP mode.

- The generation of the wake-up factors can be determined by the wake-up factor flags (WUF0).
- The ports in the Isolated area maintain the I/O buffer hold state.
  Release the I/O buffer hold state by executing the following steps:

  1. Re-configure the peripheral functions and port functions.

  2. Set IOHOLD.IOHOLD = 0.

- To execute an interrupt of the wake-up factor after the wake-up, evaluate the information of wake-up factor flag by software and set the interrupt request flag in the interrupt control register. Then, when an interrupt is enabled by the CPU instruction “EI”, the generated wake-up interrupt will be executed.




Note 1. When the operation of the peripheral function is stopped during operating due to the transition to the DeepSTOP mode, the operation of the peripheral function may be incorrect. Before the transition to the DeepSTOP mode, stop all of the peripheral functions to which the clock supply is to be stopped.

Note 2. After setting STBC0PSC = 0000 $0002_H$, wait for the transition to the DeepSTOP mode by the unconditional loop.

Note 3. The CPU starts the program from the reset vector after the generation of a wake-up factor. The return from the DeepSTOP mode by a reset can be checked by the RESF register. In addition, the generated wake-up factor can be checked by the WUF0 register.

Note 4. This processing is optional. It is required when executing the interrupt handling after the wake-up.

**Figure 14.3 Example of DeepSTOP Mode Transition**

### 14.3.3 Cyclic RUN Mode

In Cyclic RUN mode, the functions except the CPU, peripheral functions in Always-On area (AWO area), RLIN3, and CSIG0 are stopped. In this mode, PLL and Flash Memory are not available. The transition procedure (example) to Cyclic RUN mode is shown below.

#### Preparation of Cyclic RUN

Allocate the program for Cyclic RUN to the Retention RAM. The reset vector base address (RBASE) in Cyclic RUN operation should be specified in the CYCRBASE register described in **Section 9. Reset controller**. Note that neither the code flash memory nor the data flash memory is available in Cyclic RUN mode.

The instruction to transition to DeepSTOP mode should be arranged in the interrupt exception handler or a polling routine of interrupt request which is used as the source of returning to the RUN mode.

For details on the exception vector, see *the RH850G3KH User's Manual: Software*.

**CAUTION**

**Do not change the PSW.EBV bit from its value after reset in Cyclic RUN mode (Do not set the PSW.EBV bit to 1 in Cyclic RUN mode).**

- Set the wake-up related registers.
  - Clear the wake-up factor flags (the WUFC20 register).
  - Mask the non-wake-up factor (the WUFMSK20 register).
  - Release the masks of the wake-up factors (the WUFMSK20 register).
- Transition to DeepSTOP mode. For details on the transition to DeepSTOP mode, see **Section 14.3.2, DeepSTOP Mode**.

#### Start of Cyclic RUN

The operation transitions to Cyclic RUN mode from DeepSTOP mode at the generation of wake-up factor 2.

The operation transitions to Cyclic RUN mode from Cyclic STOP mode at the generation of wake-up factors 1 and 2.

The microcontroller starts operation from the reset vector address of Cyclic RUN mode specified by the CYCRBASE register. If one of the following interrupts has been generated during recovery from DeepSTOP mode to Cyclic RUN mode, the microcontroller restart operation from the exception handler address:

- FENMI: FENMI handler address in Cyclic RUN mode (CYCRBASE + $E0_H$)
- FEINT: FEINT handler address in Cyclic RUN mode (CYCRBASE + $F0_H$)

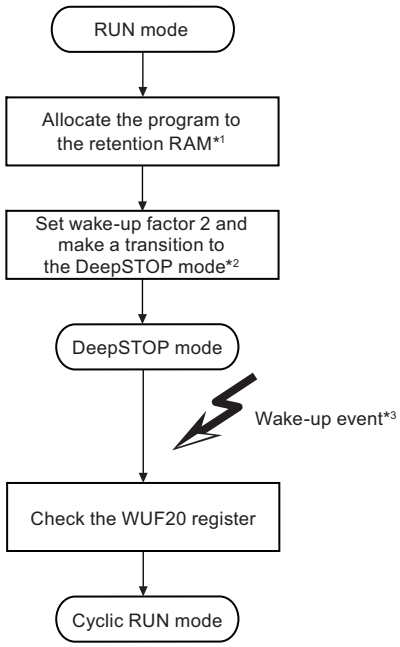
Note that the General-purpose registers and Local RAM are undefined value after the transition to Cyclic RUN mode from DeepSTOP mode.

#### End of Cyclic RUN

The Cyclic RUN mode ends at the transition to the Cyclic STOP mode by setting the STBC0STPT.STBC0STPTRG bit to 1, or at the transition to the DeepSTOP mode by setting the STBC0PSC.STBC0DISTRG bit to 1.

### Wake-up handling

The generation of the wake-up factors can be determined by the wake-up factor flags (WUF20).




Note 1. When the mode transitions from the Cyclic RUN mode to the RUN mode by a wake-up factor, the transition to the DeepSTOP mode should be made in the processing of the interrupt vector for the wake-up factor. In that case, allocate the interrupt processing program to the retention RAM as well.

Note 2. Before the transition to the DeepSTOP mode, clear the flag for wake-up factor 2 in the WUFC20 register and set wake-up factor 2 that is to be used in the WUFMSK20 register. Other parts of the procedure for transition to DeepSTOP mode are the same as the normal procedure.

Note 3. The CPU starts the program from the address specified by CYCRBASE register after the generation of a wake-up factor. The generated wake-up factor can be checked by the WUF20 register.

**Figure 14.4 Example of Cyclic RUN Mode Transition**

### 14.3.4 Cyclic STOP Mode

In Cyclic STOP mode, the functions except the peripheral functions in the Always-On area (AWO area) and RLIN3 are stopped.

The transition procedure (example) to Cyclic STOP mode is shown below.

#### Preparation for Cyclic STOP

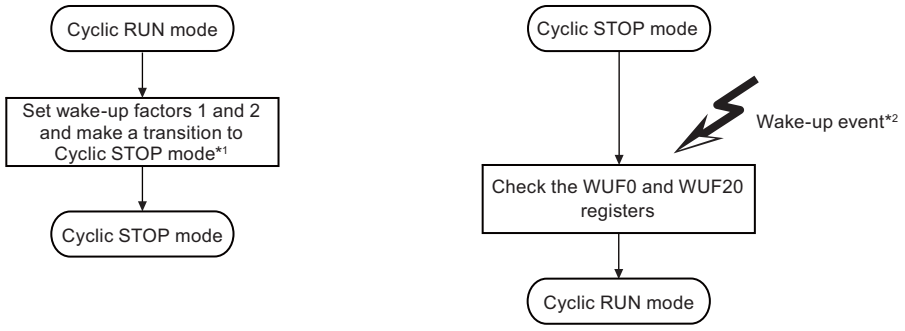
- Transition to Cyclic RUN mode.
- Set the wake-up related registers.
  - Clear the wake-up factor flags (the WUFC0/WUFC20 register).
  - Mask the non-wake-up factor (the WUFMSK0/WUFMSK20 register).
  - Release the masks of the wake-up factors (the WUFMSK0/WUFMSK20 register).

#### Start of Cyclic STOP

Set the STBC0STPT.STBC0STPTR bit to 1 to transition to Cyclic STOP mode.

#### End of Cyclic STOP

The operation transitions to Cyclic RUN mode at the generation of wake-up factor 1 or 2.




Note 1. Set wake-up factors 1 and 2 to be used for transitioning to RUN mode and Cyclic RUN mode, respectively. When the mode transitions to RUN mode by wake-up factor 1, the transition processing to DeepSTOP mode should be added in Cyclic RUN mode.

Note 2. When a wake-up factor is generated in Cyclic STOP mode, the mode transitions to Cyclic RUN mode and the operation starts immediately after the processing shifted to Cyclic STOP mode. The generated wake-up factors can be checked by the WUF0 and WUF20 registers.

**Figure 14.5 Example of Cyclic STOP Mode Transition**

## 14.4 Writing to the Stand-By Controller Related Registers

The following stand-by controller registers are write-protected registers.

- STBC0PSC register
- STBC0STPT register
- IOHOLD register

The write-protected registers are protected against the illegal writing due to an incorrect program operation.

For details on the write-protected sequence, see **Section 5, Write-Protected Registers**.

## 14.5 Clock Oscillator Behavior During Stand-By Mode Transition

The following figures explain clock oscillator behavior during stand-by mode transition. The clock oscillators restart operation automatically if they are used before entering stand-by.

- If MainOSC and PLL are enabled before entering STOP mode, they restart oscillation automatically during wake-up from STOP mode, and CPU restarts operation after oscillations of these clock sources become stable.
- If MainOSC is enabled before entering DeepSTOP mode, it restarts oscillation automatically during wake-up from DeepSTOP mode, and CPU restarts operation after oscillation of MainOSC becomes stable. PLL is not restarted automatically even if it is enabled before entering DeepSTOP mode.
- If MainOSC is enabled before entering Cyclic STOP mode, it restarts oscillation automatically during wake-up from Cyclic STOP mode. CPU restarts operation after oscillation of MainOSC becomes stable.

Note that behavior of HS IntOSC and MainOSC in the following figures is in the case they stop oscillation during stand-by mode. HS IntOSC and MainOSC continue oscillation during stand-by mode if their stop mask register is set to 1 (ROSCSTPMSK bit of ROSCSTPM register and MOSCSTPMSK bit of MOSCSTPM register respectively) or there is a clock domain which uses the source clock during stand-by by setting its stop mask bit (CKSC_xxxx_STPM = $0000_0003_H$).

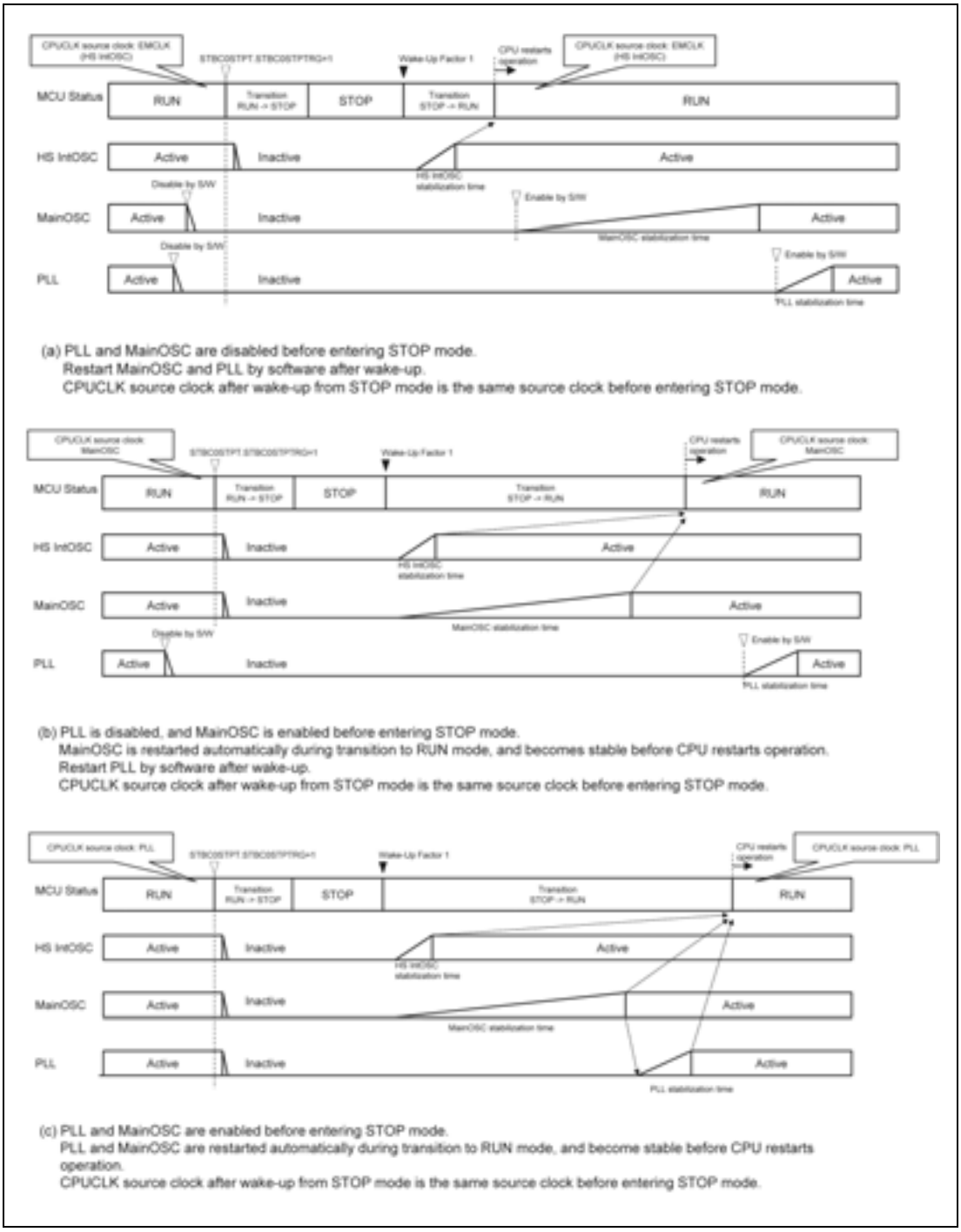



**Figure 14.6 Clock oscillators behavior in stand-by mode transition (RUN → STOP → RUN)**

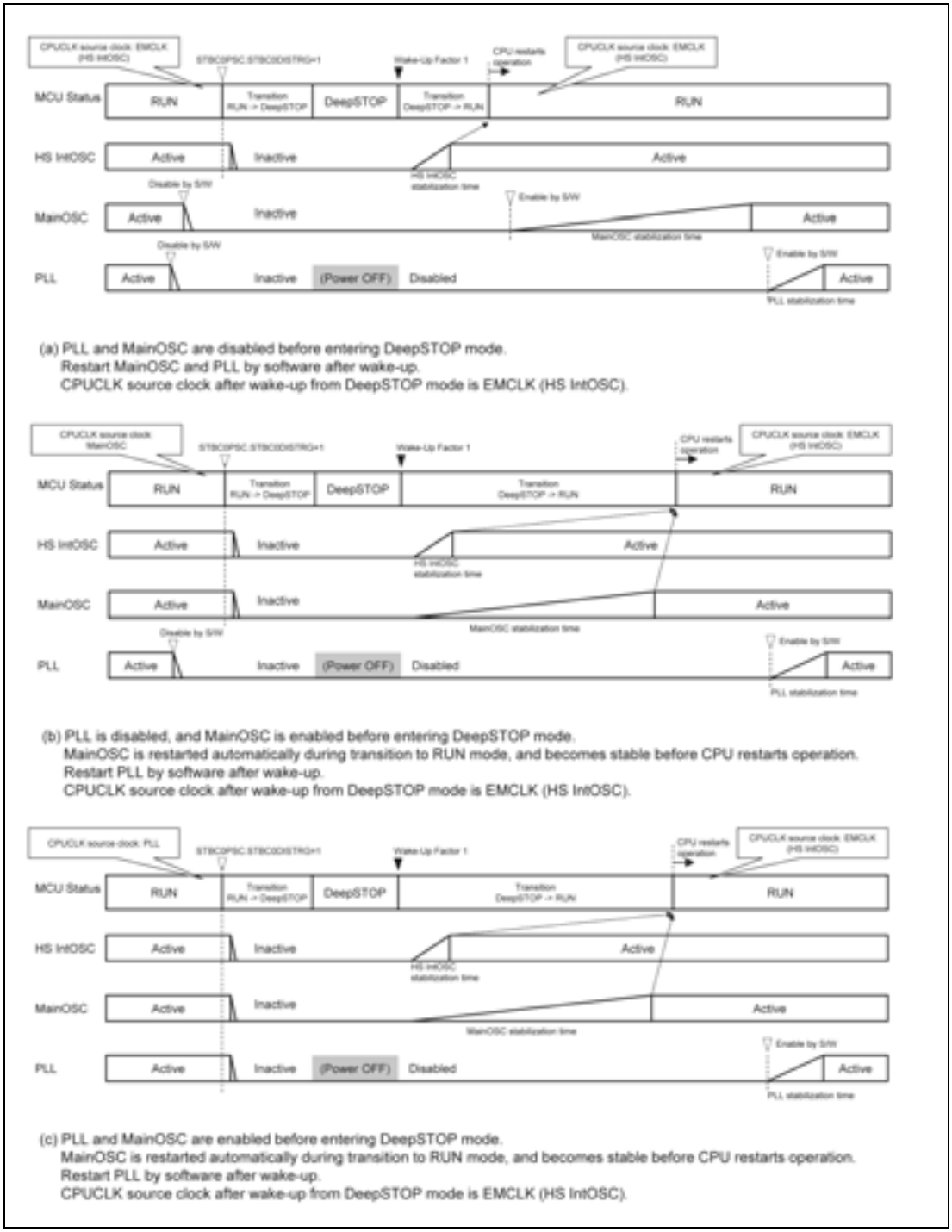



**Figure 14.7 Clock oscillators behavior in stand-by mode transition (RUN → DeepSTOP → RUN)**

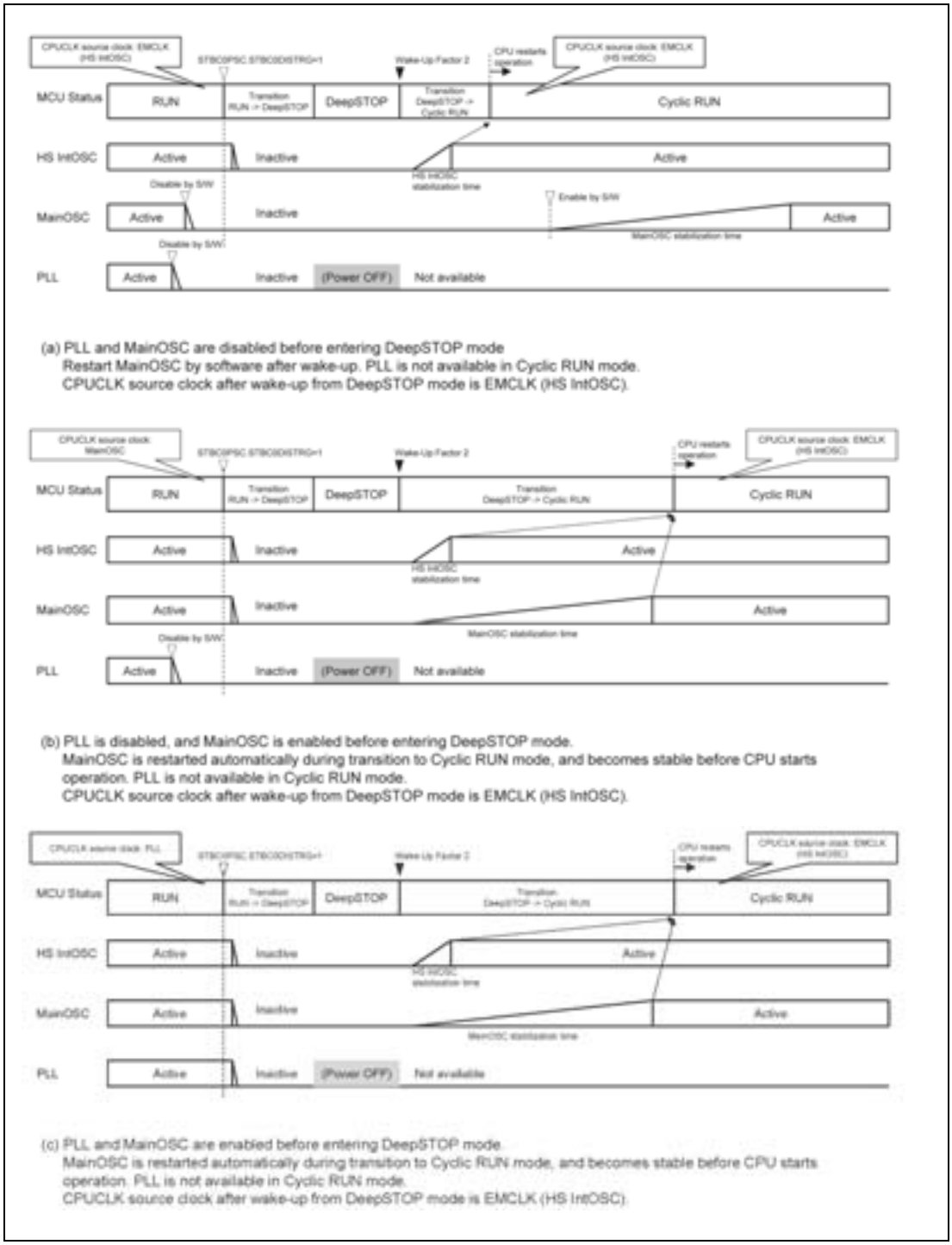



**Figure 14.8 Clock oscillators behavior in stand-by mode transition (RUN → DeepSTOP → Cyclic RUN)**

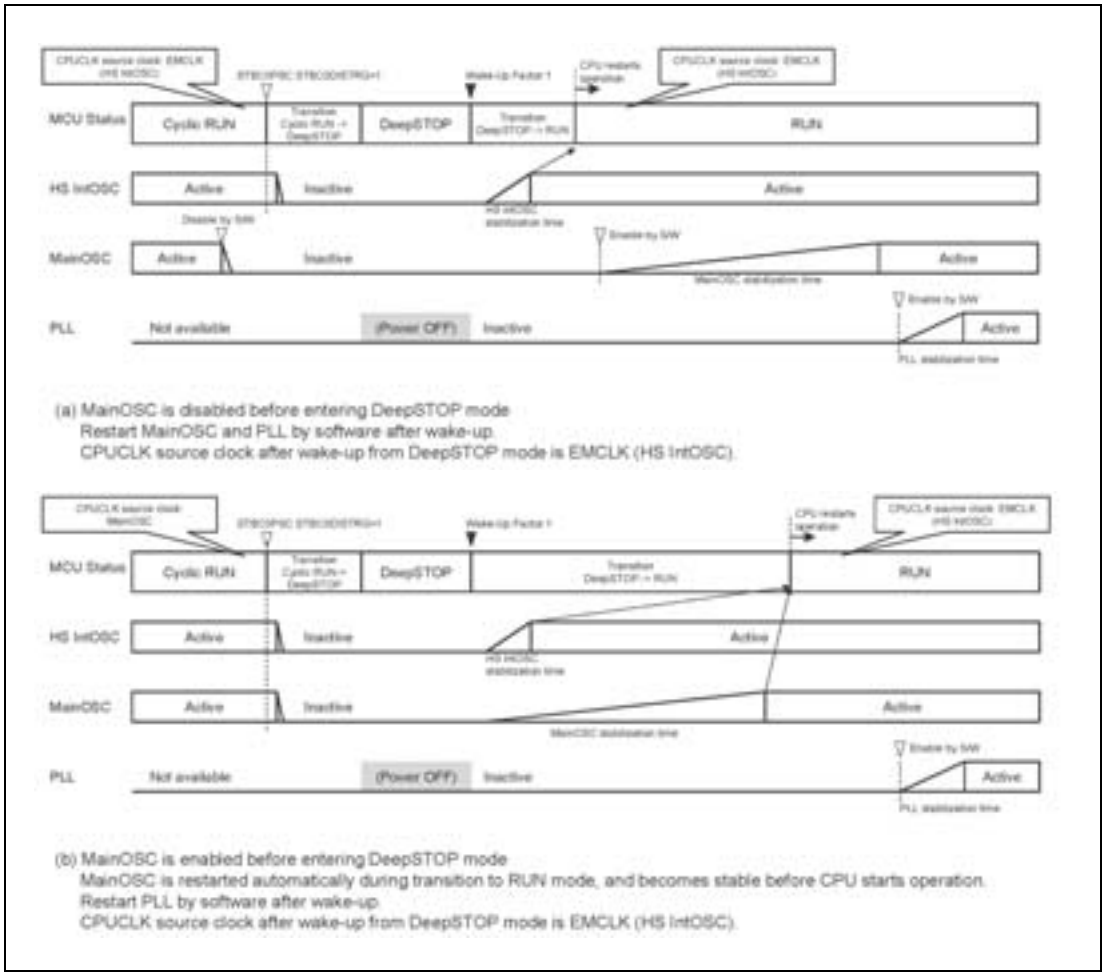



**Figure 14.9 Clock oscillators behavior in stand-by mode transition (Cyclic RUN → DeepSTOP → RUN)**

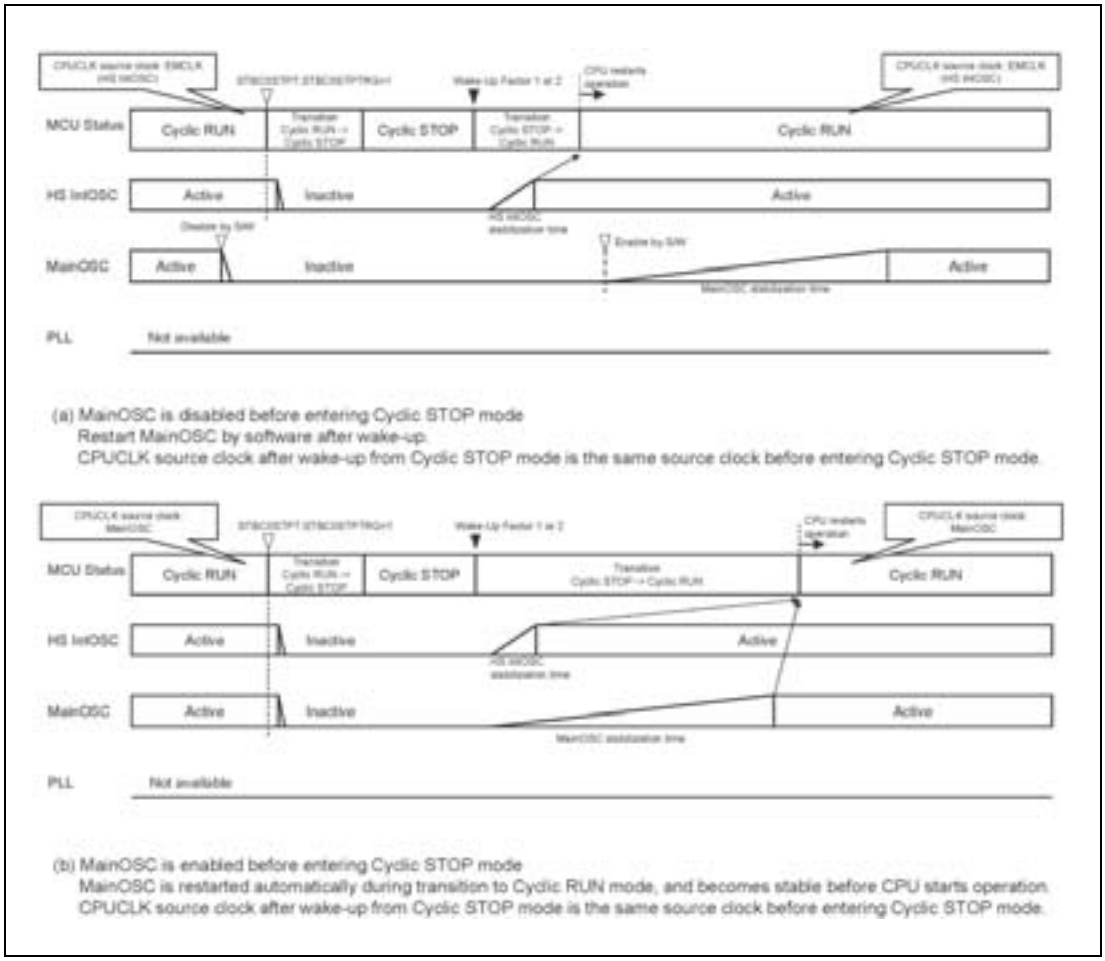



**Figure 14.10 Clock oscillators behavior in stand-by mode transition (Cyclic RUN → Cyclic STOP → Cyclic RUN)**

## 14.6 Cautions when Using Stand-By Modes

### 14.6.1 Cautions Concerning Transitioning to DeepSTOP Mode When Using a Debugger

When using a debugger, executing a program that causes the mode to transition to DeepSTOP mode immediately after the program is started may cause improper communication between the OCD emulator and microcontroller because the microcontroller will enter DeepSTOP mode before the preparations for communication between the OCD emulator and microcontroller are completed.

The communication preparation period depends on the OCD emulator's host PC environment and the operating frequency of the microcontroller, so when performing debugging that causes the program to enter DeepSTOP mode immediately after the program starts, insert a wait between reset release and the DeepSTOP execution instruction so that the debugger starts normally.

In DeepSTOP mode, the debugging controller stops. For return from DeepSTOP mode by the debugger, see **Section 14.1.3, On-Chip Debug Wake-Up**.

# Section 15 Low-Power Sampler (LPS)

This section contains a generic description of the low-power sampler (LPS).

The first part in this section describes all RH850/F1K specific properties, such as the number of units, register base addresses, etc. The remainder of the section describes the functions and registers of the LPS.

## 15.1 Features of RH850/F1K LPS

### 15.1.1 Number of Units

This microcontroller has the following number of LPS units.

**Table 15.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of Units | 1 | 1 | 1 |
| Name | LPSn (n = 0) | LPSn (n = 0) | LPSn (n = 0) |

**Table 15.2 Unit Configurations and Channels**

| Unit Name LPSn | Channels per Unit | Function | Channel Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|---|---|---|
| LPS0 | 1 | Digital port input m for port polling | DPINm | 17 ch | 24 ch | 24 ch |
| | | Analog input m for A/D converter | ADCA0Im | 16 ch | 16 ch | 16 ch |

**Table 15.3 Indices**

| Index | Description |
|---|---|
| n | Throughout this section, the individual LPS units are identified by the index “n” (n = 0). |
| m | Throughout this section, the number of digital port input channels for LPS port polling is indicated by the index “m” (m = 0 to 23) and the number of analog input channels for A/D converter is indicated by the index “m” (m = 0 to 15) |
| k | The external multiplexer select output signal is indicated by the index “k”. |
| x | LPS sequence start trigger input signal is indicated by the index “x”. |

**NOTE**

Descriptions of functions and registers in this section are based on the maximum configurations. Adjust the indices in the text to the proper value for each product. When writing a value to a register that will result in writing to bits outside the range of the index for the product you are using, write the value after reset to these bits.

The following table shows values indicated by the indices of each product.

**Table 15.4 Indices of Products**

| Indices of Each Product |
|---|
| 100 pins, 144 pins, 176 pins |
| k = 0 to 2 |
| x = 0 to 3 |

### 15.1.2 Register Base Address

The LPS base address is shown in the following table.

LPS register addresses are given as offsets from the base addresses.

**Table 15.5 Register Base Address**

| Base Address Name | Base Address |
|---|---|
| <LPS0_base> | FFF8 $3000_H$ |

### 15.1.3 Clock Supply

The LPS clock supply is shown in the following table.

If the operation request signal for the low-power sampler (LPS) is at the active level, the clock for clock domains for which the HS IntOSC is selected also operates.

To stop the function of a clock domain, set the target clock domain to “disabled” before making a transition to standby mode.

**Table 15.6 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name |
|---|---|---|
| LPSn | Register access clock | CPUCLK2 |
| | | EMCLK |
| | Operating clock | EMCLK |

### 15.1.4 Interrupt Request

The LPS interrupt requests are listed in the following table.

**Table 15.7 Interrupt Requests**

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **LPS0** | | | |
| INTCWEND | Port polling end interrupt (LPS) | 112 | — |
| INTDPE | Digital port error interrupt (LPS) | 356 | — |
| INTAPE | Analog port error interrupt (LPS) | 357 | — |
| INTADCA0I0*1 | ADCA0 SG1 end interrupt | 18 | 4 |
| INTADCA0I1*1 | ADCA0 SG2 end interrupt | 19 | 5 |
| INTADCA0I2*1 | ADCA0 SG3 end interrupt | 20, 32 | 6 |

Note 1. These signals are output from ADCA0.

### 15.1.5 Reset Sources

The LPS reset sources are shown in the following table. The LPS is initialized by these reset sources.

**Table 15.8 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| LPS0 | All reset sources except transition to DeepSTOP mode (AWORES) |

### 15.1.6 External Input/Output Signals

External input/output signals of LPS are listed below.

**Table 15.9 External Input/Output Signals**

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **LPS0** | | |
| DPO | Port output signal for digital input | DPO |
| DPSELk | External multiplexer select output signal for digital port | SELDPk |
| DPINm | Digital port input signal | DPINm |
| APO | Port output signal for analog input | APO |
| ADCA0SELk*1 | External analog multiplexer (MPX) output pin | ADCA0SELk |
| ADCA0Im*1 | ADCA input channel signal | ADCA0Im |

Note 1. These signals are input/output of ADCA0. For details, see Section 31, A/D Converter (ADCA).

### 15.1.7 Internal Input/Output Signals

Internal input/output signals for connecting the LPS and the STBC or the LPS and the TAUJ are listed below.

**Table 15.10 Internal Input/Output Signals**

| Unit Signal Name | Description | Connected to |
|---|---|---|
| WUTRG0 | LPS wake-up source trigger 0 output signal | STBC |
| WUTRG1 | LPS wake-up source trigger 1 output signal | STBC |
| INTTAUJ0Ix | LPS sequence start trigger x input signal | TAUJ0 |

## 15.2 Overview

### 15.2.1 Functional Overview

To monitor the external input without consuming CPU resources, the low-power sampler (LPS) can check the digital input ports and analog input ports without using the CPU. The figure below shows a connection example between the main components of the LPS and the external circuit.

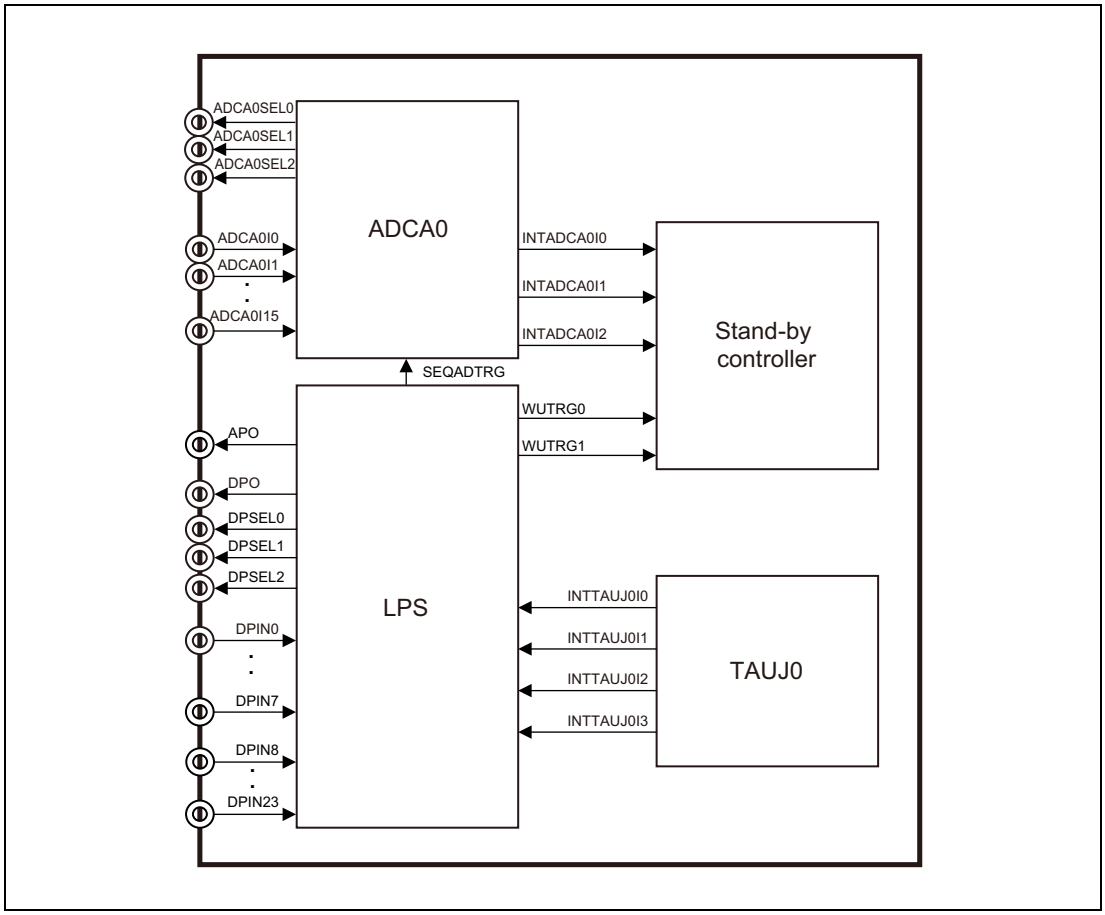



**Figure 15.1 Block Diagram of the LPS**

**CAUTION**

**DPSEL2 to DPSEL0 are assigned to the same pins as DPIN10 to DPIN8 as alternative-function. They cannot be used simultaneously.**

# 15.3 Registers

## 15.3.1 List of Registers

LPS registers are listed in the following table.

For details about <LPS0_base>, see **Section 15.1.2, Register Base Address**.

**Table 15.11 List of Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| LPS0 | LPS control register | SCTLR | <LPS0_base > + $00_H$ |
| | Event flag register | EVFR | <LPS0_base > + $04_H$ |
| | DPIN select register 0 | DPSELR0 | <LPS0_base > + $08_H$ |
| | DPIN select register M | DPSELRM | <LPS0_base > + $0C_H$ |
| | DPIN select register H | DPSELRH | <LPS0_base > + $10_H$ |
| | DPIN data set register 0 | DPDSR0 | <LPS0_base > + $14_H$ |
| | DPIN data set register M | DPDSRM | <LPS0_base > + $18_H$ |
| | DPIN data set register H | DPDSRH | <LPS0_base > + $1C_H$ |
| | DPIN data input monitor register 0 | DPDIMR0 | <LPS0_base > + $20_H$ |
| | DPIN data input monitor register 1 | DPDIMR1 | <LPS0_base > + $24_H$ |
| | DPIN data input monitor register 2 | DPDIMR2 | <LPS0_base > + $28_H$ |
| | DPIN data input monitor register 3 | DPDIMR3 | <LPS0_base > + $2C_H$ |
| | DPIN data input monitor register 4 | DPDIMR4 | <LPS0_base > + $30_H$ |
| | DPIN data input monitor register 5 | DPDIMR5 | <LPS0_base > + $34_H$ |
| | DPIN data input monitor register 6 | DPDIMR6 | <LPS0_base > + $38_H$ |
| | DPIN data input monitor register 7 | DPDIMR7 | <LPS0_base > + $3C_H$ |
| | Count value register | CNTVAL | <LPS0_base > + $40_H$ |
| | LPS operation status register | SOSTR | <LPS0_base > + $44_H$ |

## 15.3.2 SCTLR — LPS Control Register

This register is used to configure the LPS.

**Access:** This register can be read or written in 32-bit units.

**Address:** <LPS0_base> + $00_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | NUMDP2 | NUMDP1 | NUMDP0 | TJIS1 | TJIS0 | ADEN | DPEN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 15.12 SCTLR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 6 to 4 | NUMDP[2:0] | These bits specify the number of times the port is read in digital input mode. If two or more times are specified, the external multiplexer is controlled by the DPSEL[2:0] pins.<br>The bits for which comparison is enabled in the DPSELR0, DPSELRM, and DPSELRH registers are compared regardless of the repeat number setting, and WUTRG will be generated according to the results.<br>**NUMDP[2:0] / Number of Times the Port Is Read**<br>$000_B$ One time<br>$001_B$ Two times<br>$010_B$ Three times<br>$011_B$ Four times<br>$100_B$ Five times<br>$101_B$ Six times<br>$110_B$ Seven times<br>$111_B$ Eight times<br>These bits should be set before the TAUJ0 and sequence operations are started (when the SCTLR.DPEN bit = 0, the SCTLR.ADEN bit = 0, and the SOSTR.SOF bit = 0).<br>(When changing the SCTLR.DPEN bit and the SCTLR.ADEN bit, write the same value to these bits.) |
| 3, 2 | TJIS[1:0] | Sequence Start Trigger Select<br>00: INTTAUJ0I0<br>01: INTTAUJ0I1<br>10: INTTAUJ0I2<br>11: INTTAUJ0I3<br>These bits should be set before the sequence operation is started (when the SCTLR.DPEN bit = 0, the SCTLR.ADEN bit = 0, and the SOSTR.SOF bit = 0).<br>(When changing the SCTLR.DPEN bit and the SCTLR.ADEN bit, write the same value to these bits.) |
| 1 | ADEN | 0: Analog input mode is disabled<br>1: Analog input mode is enabled |

Table 15.12 SCTLR Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | DPEN | 0: Digital input mode is disabled<br>1: Digital input mode is enabled |

### 15.3.3 EVFR — Event Flag Register

This register indicates the result of comparing the data sequentially captured at the digital input pins and stored in the DPDIMR7 to DPDIMR0 registers with the comparison target data in the DPDSRH/DPDSRM/DPDSR0 registers.

**Access:** This register can be read or written in 32-bit units.

**Address:** <LPS0_base> + $04_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DINEVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 15.13 EVFR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | DINEVF | This bit indicates the result of comparing the data captured at the digital input pins and stored in the DPDIMR7 to DPDIMR0 registers with the comparison target data in the DPDSRH/DPDSRM/DPDSR0 registers.<br>Read:<br>0: The result of comparison is a match.<br>1: The result of comparison is a mismatch.<br>Write:<br>0: Clear the bit.<br>1: Prohibited.<br>This bit is set to 1 when a mismatch is detected even in one bit. Only 0 can be written to clear this bit. |

### 15.3.4 DPSELR0 — DPIN Select Register 0

This register specifies the compare target bits in the DPDSR0 and DPDIMR0 registers.

Write to the DPSELR0 register before the sequence operation is started (when the SOSTR.SOF bit = 0).

**Access:** This register can be read or written in 32-bit units.

**Address:** <LPS0_base> + $08_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | D0EN_23 | D0EN_22 | D0EN_21 | D0EN_20 | D0EN_19 | D0EN_18 | D0EN_17 | D0EN_16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D0EN_15 | D0EN_14 | D0EN_13 | D0EN_12 | D0EN_11 | D0EN_10 | D0EN_9 | D0EN_8 | D0EN_7 | D0EN_6 | D0EN_5 | D0EN_4 | D0EN_3 | D0EN_2 | D0EN_1 | D0EN_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 15.14 DPSELR0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 23 to 0 | D0EN_n<br>(n = 23 to 0) | These bits enable or disable comparing each bit of the first data captured at the digital input pins and stored in the DPDIMR0 register with the comparison target data in the DPDSR0 register.<br>0: Disables comparison.<br>1: Enables comparison. |

### 15.3.5 DPSELRM — DPIN Select Register M

This register specifies the compare target bits in the DPDSRM and DPDIMRm (m = 4 to 1) registers.

Write to the DPSELRM register before the sequence operation is started (when the SOSTR.SOF bit = 0).

**Access:** DPSELRM can be read or written in 32-bit units.
DPSELRML and DPSELRMH can be read or written in 16-bit units.
DPSELR1, DPSELR2, DPSELR3, and DPSELR4 can be read or written in 8-bit units.

**Address:** DPSELRM: <LPS0_base> + $0C_H$
DPSELRML: <LPS0_base> + $0C_H$,
DPSELRMH: <LPS0_base> + $0E_H$
DPSELR1: <LPS0_base> + $0C_H$,
DPSELR2: <LPS0_base> + $0D_H$,
DPSELR3: <LPS0_base> + $0E_H$,
DPSELR4: <LPS0_base> + $0F_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D4EN_7 | D4EN_6 | D4EN_5 | D4EN_4 | D4EN_3 | D4EN_2 | D4EN_1 | D4EN_0 | D3EN_7 | D3EN_6 | D3EN_5 | D3EN_4 | D3EN_3 | D3EN_2 | D3EN_1 | D3EN_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D2EN_7 | D2EN_6 | D2EN_5 | D2EN_4 | D2EN_3 | D2EN_2 | D2EN_1 | D2EN_0 | D1EN_7 | D1EN_6 | D1EN_5 | D1EN_4 | D1EN_3 | D1EN_2 | D1EN_1 | D1EN_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 15.15 DPSELRM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | D4EN_n (n = 7 to 0) | These bits enable or disable comparing each bit of the fifth data captured at the digital input pins and stored in the DPDIMR4 register with the comparison target data in the DPDSR4 register.<br>0: Disables comparison.<br>1: Enables comparison. |
| 23 to 16 | D3EN_n (n = 7 to 0) | These bits enable or disable comparing each bit of the fourth data captured at the digital input pins and stored in the DPDIMR3 register with the comparison target data in the DPDSR3 register.<br>0: Disables comparison.<br>1: Enables comparison. |
| 15 to 8 | D2EN_n (n = 7 to 0) | These bits enable or disable comparing each bit of the third data captured at the digital input pins and stored in the DPDIMR2 register with the comparison target data in the DPDSR2 register.<br>0: Disables comparison.<br>1: Enables comparison. |
| 7 to 0 | D1EN_n (n = 7 to 0) | These bits enable or disable comparing each bit of the second data captured at the digital input pins and stored in the DPDIMR1 register with the comparison target data in the DPDSR1 register.<br>0: Disables comparison.<br>1: Enables comparison. |

### 15.3.6 DPSELRH — DPIN Select Register H

This register specifies the compare target bits in the DPDSRH and DPDIMRm (m = 7 to 5) registers.

Write to the DPSELRH register before the sequence operation is started (when the SOSTR.SOF bit = 0).

**Access:** DPSELRH can be read or written in 32-bit units.
DPSELRHL and DPSELRHH can be read or written in 16-bit units.
DPSELR5, DPSELR6, and DPSELR7 can be read or written in 8-bit units.

**Address:** DPSELRH: <LPS0_base> + $10_H$
DPSELRHL: <LPS0_base> + $10_H$,
DPSELRHH: <LPS0_base> + $12_H$
DPSELR5: <LPS0_base> + $10_H$,
DPSELR6: <LPS0_base> + $11_H$,
DPSELR7: <LPS0_base> + $12_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | D7EN_7 | D7EN_6 | D7EN_5 | D7EN_4 | D7EN_3 | D7EN_2 | D7EN_1 | D7EN_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D6EN_7 | D6EN_6 | D6EN_5 | D6EN_4 | D6EN_3 | D6EN_2 | D6EN_1 | D6EN_0 | D5EN_7 | D5EN_6 | D5EN_5 | D5EN_4 | D5EN_3 | D5EN_2 | D5EN_1 | D5EN_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 15.16 DPSELRH Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 23 to 16 | D7EN_n<br>(n = 7 to 0) | These bits enable or disable comparing each bit of the eighth data captured at the digital input pins and stored in the DPDIMR7 register with the compare target data in the DPDSR7 register.<br>0: Disables comparison.<br>1: Enables comparison. |
| 15 to 8 | D6EN_n<br>(n = 7 to 0) | These bits enable or disable comparing each bit of the seventh data captured at the digital input pins and stored in the DPDIMR6 register with the compare target data in the DPDSR6 register.<br>0: Disables comparison.<br>1: Enables comparison. |
| 7 to 0 | D5EN_n<br>(n = 7 to 0) | These bits enable or disable comparing each bit of the sixth data captured at the digital input pins and stored in the DPDIMR5 register with the compare target data in the DPDSR5 register.<br>0: Disables comparison.<br>1: Enables comparison. |

### 15.3.7 DPDSR0 — DPIN Data Set Register 0

This register specifies the data to be compared with the data captured at a digital input pin and stored in the DPDIMR0 register.

Write to the DPDSR0 register before the sequence operation is started (when the SOSTR.SOF bit = 0).

**Access:** This register can be read or written in 32-bit units.

**Address:** <LPS0_base> + $14_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | D0_23 | D0_22 | D0_21 | D0_20 | D0_19 | D0_18 | D0_17 | D0_16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D0_15 | D0_14 | D0_13 | D0_12 | D0_11 | D0_10 | D0_9 | D0_8 | D0_7 | D0_6 | D0_5 | D0_4 | D0_3 | D0_2 | D0_1 | D0_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 15.17 DPDSR0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 23 to 0 | D0_n<br>(n = 23 to 0) | Data to be compared with the first digital port input (DPINm) |

### 15.3.8 DPDSRM — DPIN Data Set Register M

This register specifies the data to be compared with the data captured at a digital input pin and stored in the DPDIMR4 to DPDIMR1 registers.

Write to the DPDSRM register before the sequence operation is started (when the SOSTR.SOF bit = 0).

**Access:** DPDSRM can be read or written in 32-bit units.
DPDSRML and DPDSRMH can be read or written in 16-bit units.
DPDSR1, DPDSR2, DPDSR3, and DPDSR4 can be read or written in 8-bit units.

**Address:** DPDSRM: <LPS0_base> + $18_H$
DPDSRML: <LPS0_base> + $18_H$,
DPDSRMH: <LPS0_base> + $1A_H$
DPDSR1: <LPS0_base> + $18_H$,
DPDSR2: <LPS0_base> + $19_H$,
DPDSR3: <LPS0_base> + $1A_H$,
DPDSR4: <LPS0_base> + $1B_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D4_7 | D4_6 | D4_5 | D4_4 | D4_3 | D4_2 | D4_1 | D4_0 | D3_7 | D3_6 | D3_5 | D3_4 | D3_3 | D3_2 | D3_1 | D3_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D2_7 | D2_6 | D2_5 | D2_4 | D2_3 | D2_2 | D2_1 | D2_0 | D1_7 | D1_6 | D1_5 | D1_4 | D1_3 | D1_2 | D1_1 | D1_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 15.18 DPDSRM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | D4_n (n = 7 to 0) | Data to be compared with the fifth digital port input (DPINm) |
| 23 to 16 | D3_n (n = 7 to 0) | Data to be compared with the fourth digital port input (DPINm) |
| 15 to 8 | D2_n (n = 7 to 0) | Data to be compared with the third digital port input (DPINm) |
| 7 to 0 | D1_n (n = 7 to 0) | Data to be compared with the second digital port input (DPINm) |

### 15.3.9 DPDSRH — DPIN Data Set Register H

This register specifies the data to be compared with the data captured at a digital input pin and stored in the DPDIMR7 to DPDIMR5 registers.

Write to the DPDSRH register before the sequence operation is started (when the SOSTR.SOF bit = 0).

**Access:** DPDSRH can be read or written in 32-bit units.
DPDSRHL and DPDSRHH can be read or written in 16-bit units.
DPDSR5, DPDSR6, and DPDSR7 can be read or written in 8-bit units.

**Address:** DPDSRH: <LPS0_base> + $1C_H$
DPDSRHL: <LPS0_base> + $1C_H$,
DPDSRHH: <LPS0_base> + $1E_H$
DPDSR5: <LPS0_base> + $1C_H$,
DPDSR6: <LPS0_base> + $1D_H$,
DPDSR7: <LPS0_base> + $1E_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | D7_7 | D7_6 | D7_5 | D7_4 | D7_3 | D7_2 | D7_1 | D7_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D6_7 | D6_6 | D6_5 | D6_4 | D6_3 | D6_2 | D6_1 | D6_0 | D5_7 | D5_6 | D5_5 | D5_4 | D5_3 | D5_2 | D5_1 | D5_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 15.19 DPDSRH Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 23 to 16 | D7_n<br>(n = 7 to 0) | Data to be compared with the eighth digital port input (DPINm) |
| 15 to 8 | D6_n<br>(n = 7 to 0) | Data to be compared with the seventh digital port input (DPINm) |
| 7 to 0 | D5_n<br>(n = 7 to 0) | Data to be compared with the sixth digital port input (DPINm) |

## 15.3.10 DPDIMR0 — DPIN Data Input Monitor Register 0

This register stores the data which the LPS acquired from the digital port input (DPINm (m = 0 to 23)) in digital input mode. DPDIMR0 stores the data acquired for the first time.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <LPS0_base> + $20_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | D0M_23 | D0M_22 | D0M_21 | D0M_20 | D0M_19 | D0M_18 | D0M_17 | D0M_16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D0M_15 | D0M_14 | D0M_13 | D0M_12 | D0M_11 | D0M_10 | D0M_9 | D0M_8 | D0M_7 | D0M_6 | D0M_5 | D0M_4 | D0M_3 | D0M_2 | D0M_1 | D0M_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 15.20 DPDIMR0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | When read, the value after reset is returned. |
| 23 to 0 | D0M_n (n = 23 to 0) | The first digital port input (DPINm) data |

## 15.3.11 DPDIMR1 — DPIN Data Input Monitor Register 1

This register stores the data which the LPS acquired from the digital port input (DPINm (m = 0 to 7)) in multiplexer mode or MIX mode. DPDIMR1 stores the data acquired for the second time.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <LPS0_base> + $24_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | D1M_7 | D1M_6 | D1M_5 | D1M_4 | D1M_3 | D1M_2 | D1M_1 | D1M_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 15.21 DPDIMR1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | D1M_n (n = 7 to 0) | The second digital port input (DPINm) data |

## 15.3.12 DPDIMR2 — DPIN Data Input Monitor Register 2

This register stores the data which the LPS acquired from the digital port input (DPINm (m = 0 to 7)) in multiplexer mode or MIX mode. DPDIMR2 stores the data acquired for the third time.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <LPS0_base> + $28_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | D2M_7 | D2M_6 | D2M_5 | D2M_4 | D2M_3 | D2M_2 | D2M_1 | D2M_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 15.22 DPDIMR2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | D2M_n (n = 7 to 0) | The third digital port input (DPINm) data |

## 15.3.13 DPDIMR3 — DPIN Data Input Monitor Register 3

This register stores the data which the LPS acquired from the digital port input (DPINm (m = 0 to 7)) in multiplexer mode or MIX mode. DPDIMR3 stores the data acquired for the fourth time.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <LPS0_base> + $2C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | D3M_7 | D3M_6 | D3M_5 | D3M_4 | D3M_3 | D3M_2 | D3M_1 | D3M_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 15.23 DPDIMR3 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | D3M_n (n = 7 to 0) | The fourth digital port input (DPINm) data |

### 15.3.14 DPDIMR4 — DPIN Data Input Monitor Register 4

This register stores the data which the LPS acquired from the digital port input (DPINm (m = 0 to 7)) in multiplexer mode or MIX mode. DPDIMR4 stores the data acquired for the fifth time.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <LPS0_base> + $30_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | D4M_7 | D4M_6 | D4M_5 | D4M_4 | D4M_3 | D4M_2 | D4M_1 | D4M_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 15.24 DPDIMR4 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | D4M_n (n = 7 to 0) | The fifth digital port input (DPINm) data |

### 15.3.15 DPDIMR5 — DPIN Data Input Monitor Register 5

This register stores the data which the LPS acquired from the digital port input (DPINm (m = 0 to 7)) in multiplexer mode or MIX mode. DPDIMR5 stores the data acquired for the sixth time.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <LPS0_base> + $34_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | D5M_7 | D5M_6 | D5M_5 | D5M_4 | D5M_3 | D5M_2 | D5M_1 | D5M_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 15.25 DPDIMR5 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | D5M_n (n = 7 to 0) | The sixth digital port input (DPINm) data |

### 15.3.16 DPDIMR6 — DPIN Data Input Monitor Register 6

This register stores the data which the LPS acquired from the digital port input (DPINm (m = 0 to 7)) in multiplexer mode or MIX mode. DPDIMR6 stores the data acquired for the seventh time.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <LPS0_base> + $38_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | D6M_7 | D6M_6 | D6M_5 | D6M_4 | D6M_3 | D6M_2 | D6M_1 | D6M_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 15.26 DPDIMR6 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | D6M_n (n = 7 to 0) | The seventh digital port input (DPINm) data |

### 15.3.17 DPDIMR7 — DPIN Data Input Monitor Register 7

This register stores the data which the LPS acquired from the digital port input (DPINm (m = 0 to 7)) in multiplexer mode. DPDIMR7 stores the data acquired for the eighth time.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <LPS0_base> + $3C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | D7M_7 | D7M_6 | D7M_5 | D7M_4 | D7M_3 | D7M_2 | D7M_1 | D7M_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 15.27 DPDIMR7 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | D7M_n (n = 7 to 0) | The eighth digital port input (DPINm) data |

### 15.3.18 CNTVAL — Count Value Register

This register specifies the stabilization time of the external circuits (digital signal source and analog signal source).

- In digital mode
  The time from when the DPO output is set to 1 to the time when the port input is acquired for the first time
- In analog mode
  The time from when the APO output is set to 1 to the time when the LPS outputs the A/D conversion trigger to the ADCA0

Write to the CNTVAL register before the sequence operation is started (when the SOSTR.SOF bit = 0).

**CAUTION**

In analog mode, make sure to secure the stabilization time longer than 1 μs for the stabilization of the A/D converter.

**Access:** This register can be read or written in 16-bit units.

**Address:** <LPS0_base> + $40_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CNT17 | CNT16 | CNT15 | CNT14 | CNT13 | CNT12 | CNT11 | CNT10 | CNT07 | CNT06 | CNT05 | CNT04 | CNT03 | CNT02 | CNT01 | CNT00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 15.28 CNTVAL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 8 | CNT1n<br>(n = 7 to 0) | These bits set the stabilization time of the external circuit (analog signal source).<br>Stabilization time = $(1/f_{RH})$ × 16 × CNT1n (set value) |
| 7 to 0 | CNT0n<br>(n = 7 to 0) | These bits set the stabilization time of the external circuit (digital signal source).<br>Stabilization time = $(1/f_{RH})$ × 16 × CNT0n (set value) |

### 15.3.19 SOSTR — LPS Operation Status Register

This register indicates the operating state of the LPS.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <LPS0_base> + $44_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | SOF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 15.29 SOSTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | SOF | LPS Operation Status Flag<br>0: Initial state before the occurrence of the start trigger<br>1: LPS operation is in progress (after the start trigger occurs)<br>If the start trigger occurs while the SOF bit is set to 1 (during the LPS operation), the start trigger is canceled. |

## 15.4 Digital Input Mode

With the digital input port DPINm and the externally connected multiplexer, up to 64 input ports can be monitored as shown in **Table 15.30, Combination of Monitored Ports**.

Port DPSELk is used to switch the external multiplexer. The DPSELk output is switched for the number of times specified in the SCTLR register.

TAUJ0 is used to set the timing to check the value input to the port.

**Table 15.30 Combination of Monitored Ports**

| Combination (Number of Ports x Number of Checks) | Ports Used | Total Number |
|---|---|---|
| Direct mode<br>When input ports are checked simultaneously without using the external multiplexer<br>Up to 24 ports × 1 | DPIN23 to DPIN0 | Up to 24 |
| Multiplexer mode<br>When input ports are checked by using a small number of pins and the external multiplexer<br>Up to 8 ports × 8 | DPIN7 to DPIN0<br>DPSEL2 to DPSEL0 | Up to 64 |
| MIX mode<br>When input ports are checked using a combination of the above two modes<br>Up to 14 ports x 1 + Up to 7 ports x 7 | DPIN7 to DPIN0<br>DPIN16 to DPIN11<br>DPSEL2 to DPSEL0*1 | Up to 63 |

Note 1. DPIN16 to DPIN11 and DPIN7 are checked only for the first time. DPIN10 to DPIN8 cannot be used because they are shared with DPSEL2 to DPSEL0.

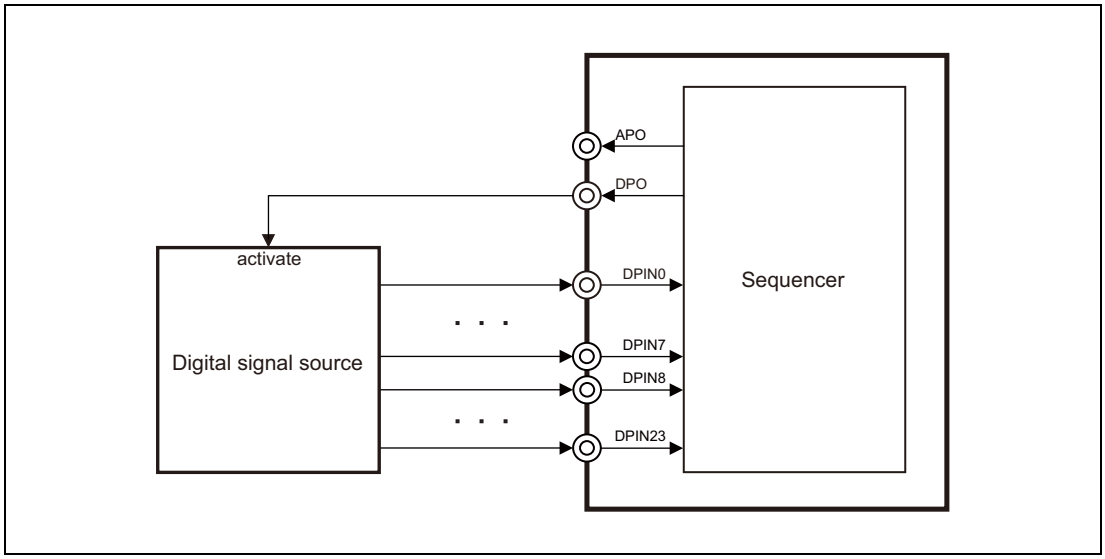



**Figure 15.2 Direct Mode Connection Example**

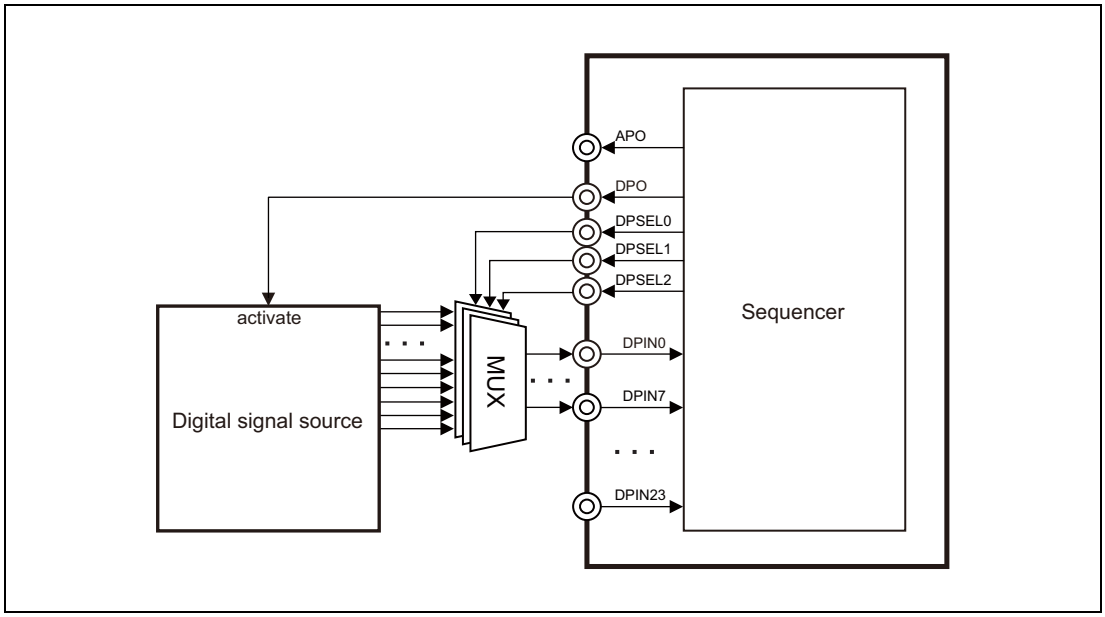



Figure 15.3 Multiplexer Mode Connection Example

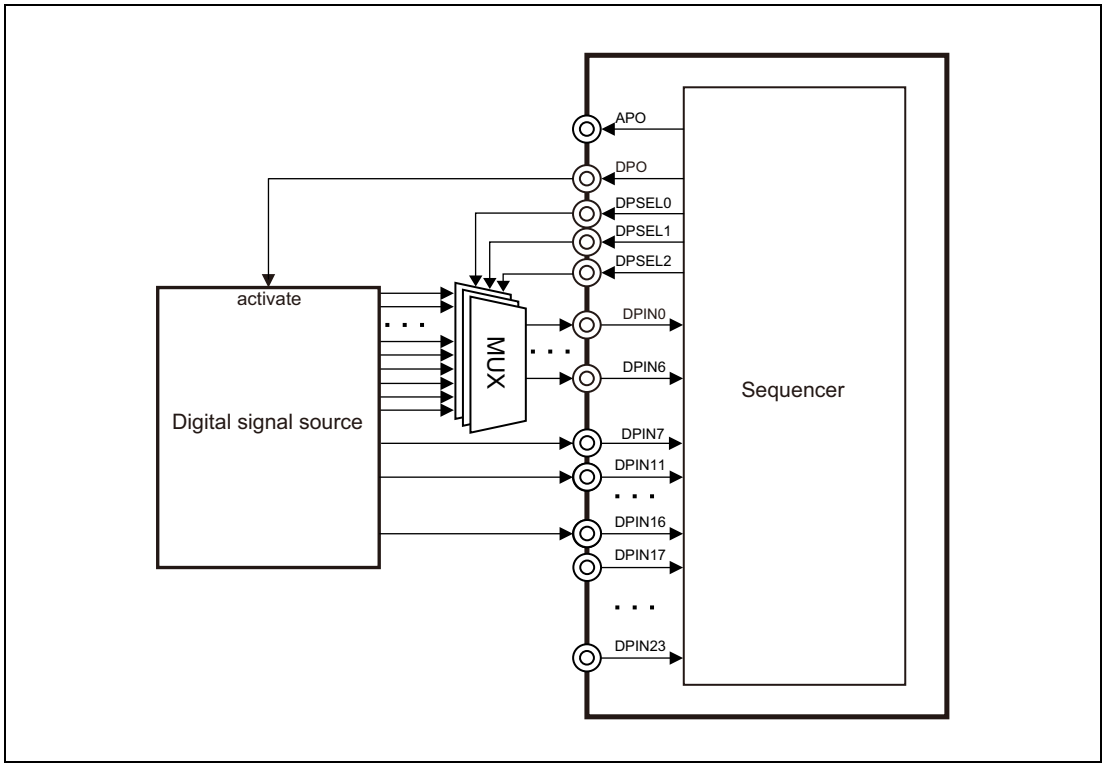



Figure 15.4 MIX Mode Connection Example

**CAUTION**

**DPSEL2 to DPSEL0 are assigned to the same pins as DPIN10 to DPIN8 as alternative-function. They cannot be used simultaneously.**

### Preparation

- Set NUMDP[2:0] and TJIS[1:0] bits in the SCTLR register to specify the number of times the port is to be read, and the TAUJ0 interrupt to be used as sequence start trigger.
- Set TAUJ0 to interval timer mode.
- Set the wait time of the digital signal source by using the lower 8 bits in the CNTVAL register.
- Set expected values in the DPDSR0, DPDSRM and DPDSRH registers.
- Set the ports to be checked in the DPSELR0, DPSELRM, and DPSELRH registers.

### Start

- Start the TAUJ0.
- Set the SCTLR.DPEN bit to 1.

After the operation starts, ports are checked at the interval set in TAUJ0. The operation continues regardless of whether the mode is RUN mode or power save mode. If the HS IntOSC is stopped in stand-by mode, it can only resume operation while the sequencer is running.

Upon completion of checking all ports that have been set, the INTCWEND interrupt occurs. In addition, if the input value of the port is different from the expected value set by the DPDSR0, DPDSRM, or DPDSRH register, the wake-up factor WUTRG0 occurs. The following figures show an example of the operation in digital input mode.

### Stop

To stop the LPS operation in Digital Input Mode (by changing the SCTLR.DPEN bit setting from 1 to 0), follow the procedure shown below. In this example, the P0_0 pin is used as DPO.

1. Set the port register to specify low level output on the pin (P0.P0_0 = 0).
2. Change the setting for the P0_0 pin from the alternative port mode to the port mode (PMC0.PMC0_0 = 0).
3. Set SCTLR.DPEN = 0.

**Note:** The above procedure applies when the P0_0 pin is used as DPO. If the P0_2 pin is used as DPO, specify the P0_2 pin settings in the same way.

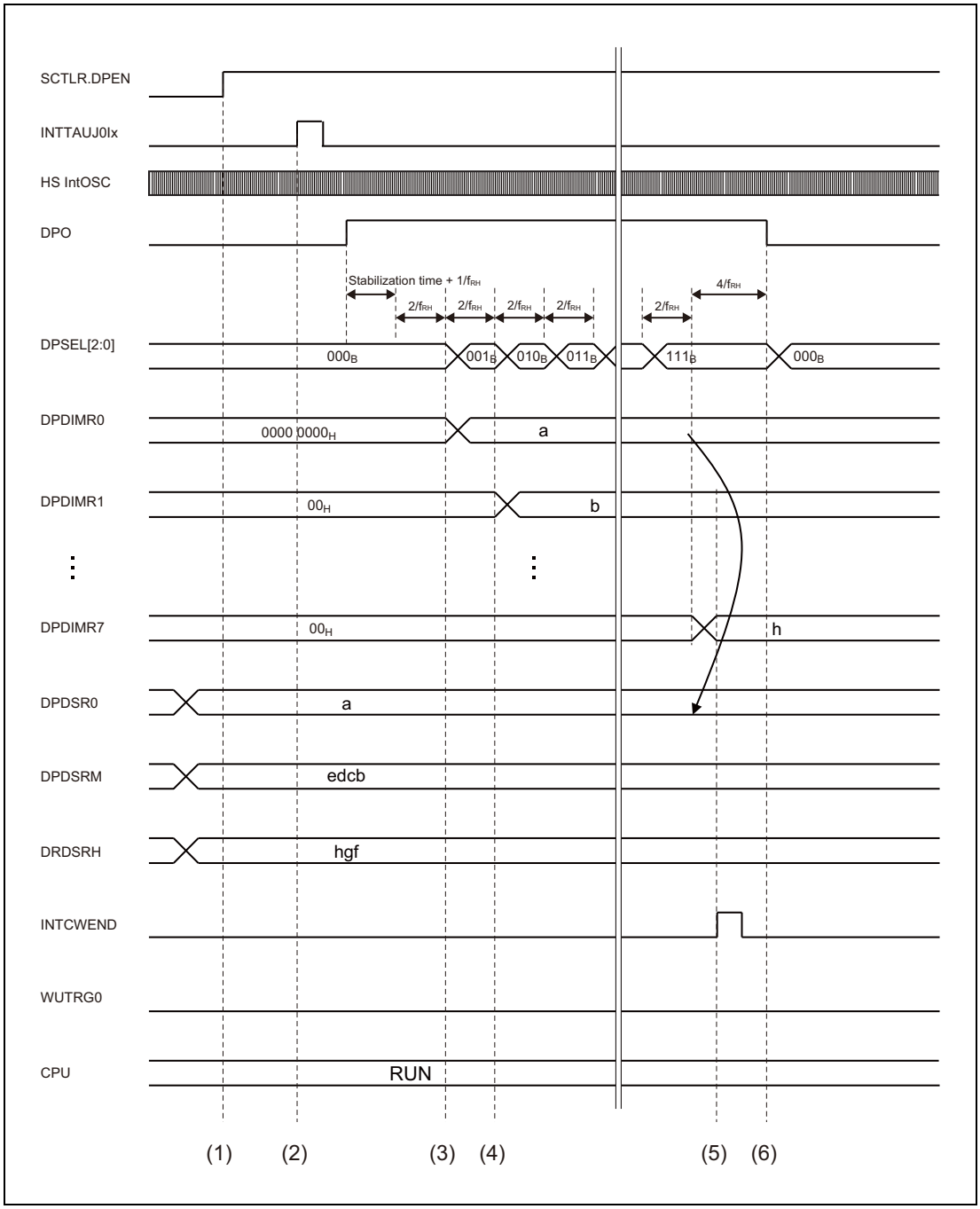



**Figure 15.5 Operation of Digital Input Mode (8 Ports × 8) when the Input Value is not Changed (RUN Mode)**

(1) Set the SCTLR.DPEN bit to 1 by software to enable the digital input mode of the LPS.

(2) When the INTTAUJ0Ix interrupt specified by the SCTLR.TJIS bit is generated, the sequencer outputs the high level from the DPO pin and waits for the time specified by CNTVAL.CNT0n to secure the stabilization of the external digital signal source.

(3) After the completion of the signal source stabilization, the LPS stores the DPIN[7:0] input value to the DPDIMR0 register and increments the DPSEL[2:0] pins to switch the external multiplexer.

(4) After the switching of the DPSEL[2:0] pins, the LPS stores the values in the DPDIMRn registers in order from DPDIMR1 and continues to increment the DPSEL[2:0] pins.

(5) After the value is stored up to the DPDIMR7 register, the INTCWEND interrupt is generated and the value is compared with the expected value set in the DPDSR0, DPDSRM, and DPDSRH registers.

(6) When the value is not different from the expected value, the wake-up factor WUTRG0 is not generated. The LPS stops the DPO output and returns to the waiting state for the trigger.

High level width of the DPO pin is calculated by the following formula. For the Stabilization time, see **Section 15.3.18, CNTVAL — Count Value Register**.

High level width of the DPO pin

= Stabilization time + $(1/f_{RH}) \times 1 + (1/f_{RH}) \times 2 \times$ (SCTLR.NUMDP (set value) + 1) + $(1/f_{RH}) \times 4$

= Stabilization time + $(1/f_{RH}) \times$ (2 × SCTLR.NUMDP (set value) + 7)

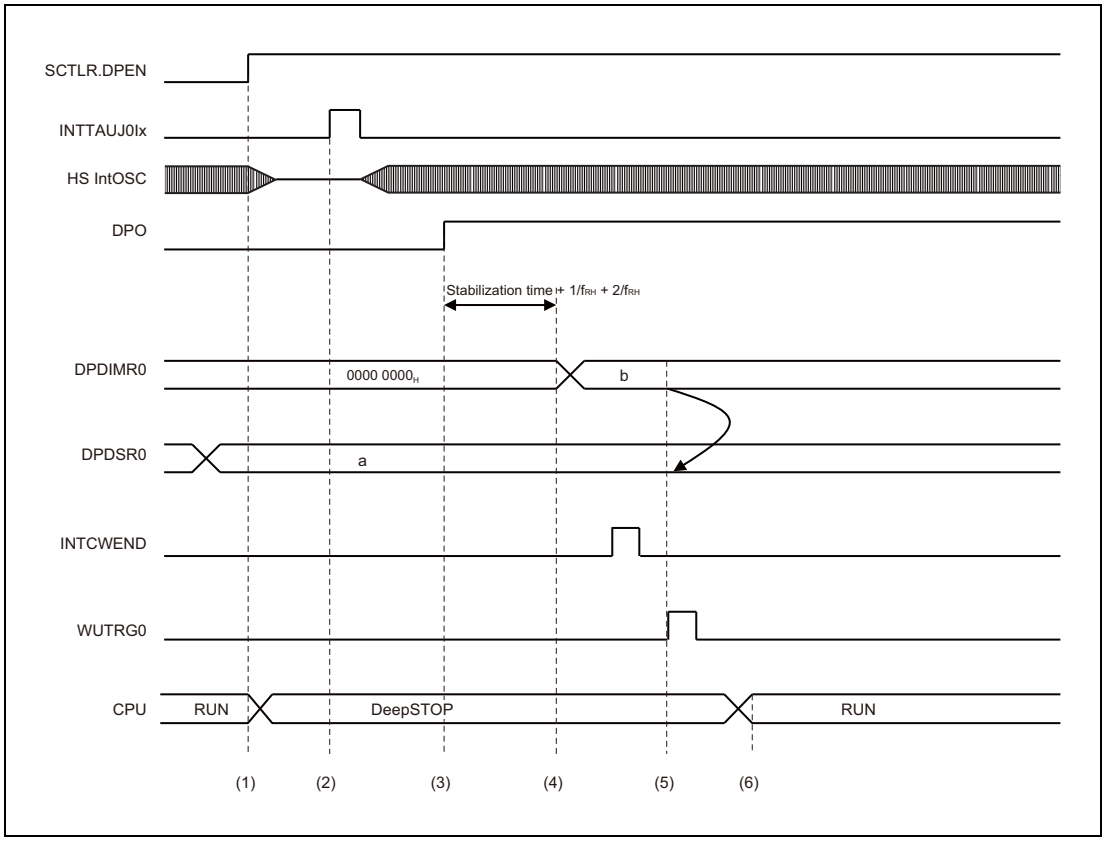



**Figure 15.6 Operation of Digital Input Mode (24 Ports × 1) when the Input Value is Changed (DeepSTOP Mode)**

(1) Set the STBC0PSC.STBC0DISTRG bit to 1 to transition to the DeepSTOP mode, while the SCTLR.DPEN bit is set to 1 by software to enable the digital input mode of the LPS.

(2) When the INTTAUJ0Ix interrupt specified by the SCTLR.TJIS bit is generated, the LPS enables the HS IntOSC to start the oscillation.

(3) After the completion of the HS IntOSC stabilization time, the LPS outputs the high level from the DPO pin and waits for the time specified by CNTVAL.CNT0n to secure the stabilization of the external digital signal source.

(4) After the completion of the signal source stabilization, the LPS stores the DPIN[23:0] input value to the DPDIMR0 register and the INTCWEND interrupt is generated.

(5) The value stored in the DPDIMR0 register is compared with the expected value set in the DPDSR0 register. When the value is different from the expected value, the wake-up factor WUTRG0 is generated.

(6) The CPU returns to RUN mode at the generation of WUTRG0. The DPO pin is driven high until the EVFR.DINEVF bit is cleared to 0 by software.

### 15.4.1 Digital Port Error Interrupt

A level sensitive interrupt indicating a data comparison mismatch is generated. This interrupt is generated not only in stand-by mode but also in RUN mode. The set and clear conditions are shown below.

**Table 15.31 Digital Port Error Interrupt**

| Unit Interrupt Signal | Set Condition | Clear Condition |
|---|---|---|
| INTDPE | When EVFR.DINEVF is set to 1 by hardware | When EVFR.DINEVF is cleared to 0 by software |

## 15.5 Analog Input Mode

The analog input port ADCA0Im (m = 0 to 15) can be monitored.

TAUJ0 is used to set the timing to check the value input to the port.

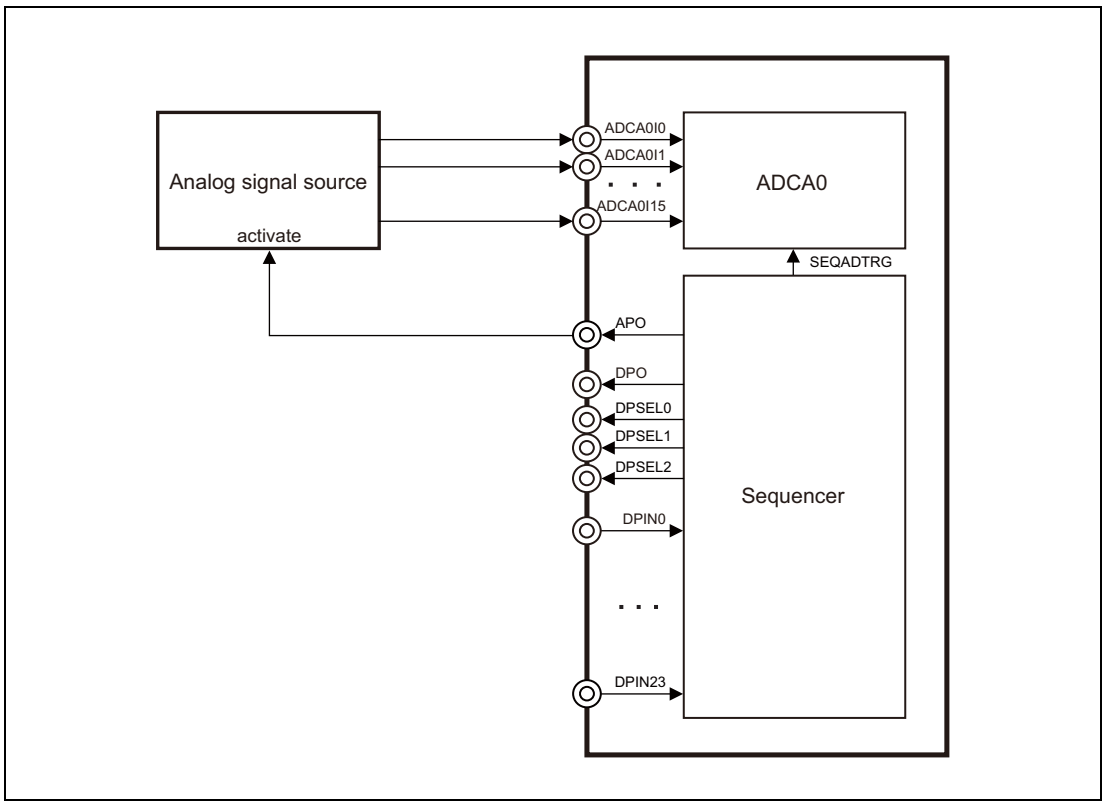



**Figure 15.7 Analog Input Mode Connection Example 1**

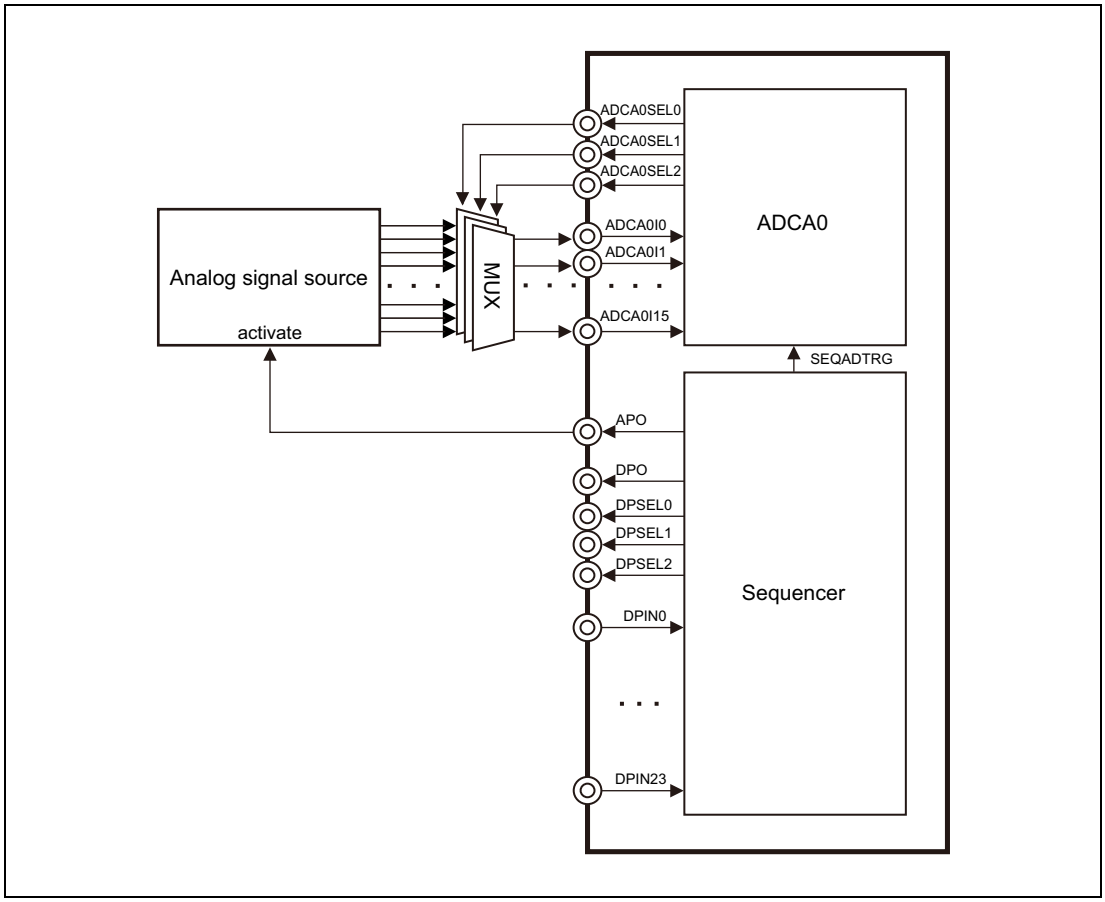



**Figure 15.8 Analog Input Mode Connection Example 2**

### Preparation

- Set TJIS[1:0] bits in the SCTLR register to specify the TAUJ0 interrupt to be used as sequence start trigger.
- Set TAUJ0 to interval timer mode.
- Set the wait time of the analog signal source by using the upper 8 bits in the CNTVAL register.
- Set the ADCA0.

### CAUTIONS

1. **When the LPS is in use, the A/D conversion completion interrupt (INT_SGx) should be output after the conversion of all channels of the LPS has been completed. The setting is as follows.**
   - **Set the ADIE bit in virtual channel register j (ADCA0VCRj) to 0 (a scan group x end interrupt (INT_SGx) is not generated when A/D conversion for virtual channel j ends in SGx.).**
   - **Set the ADIE bit in the scan group x control register (ADCA0SGCRx) to 1 (INT_SGx is output when the scan for SGx ends).**
2. **Over the period from the generation of the LPS sequence start trigger set by the SCLTR.TJIS[1:0] bits to the completion of A/D conversion for all channels of the LPS, only proceed with A/D conversion for PWM-Diag.**
   **LPS acknowledges the A/D conversion completion interrupts of the scan group**

**SG1, SG2 or SG3, but does not recognize the kind of SG (three interrupts are ORed to one interrupt). During LPS operation using one SG, an A/D conversion completion interrupt and A/D error interrupt cannot be set for another SG.**

3. **Over the period from the generation of the LPS sequence start trigger set by the SCLTR.TJIS[1:0] bits to the completion of A/D conversion for all channels of LPS, do not forcibly end A/D conversion by using the ADCA0ADHALTR.HALT bit.**
4. **When the LPS is in use, do not use the following modes.**
   - **Continuous scan mode (the setting ADCA0SGCRx.SCANMD = 1 is prohibited)**
   - **Multicycle scan mode with 2 or more cycles (the settings ADCA0SGMCYCRx.MCYC = $01_B$ and $11_B$ are prohibited)**
   - **Channel repeat mode with 2 or more cycles (the settings ADCA0SGCRx.SCT = $01_B$ and $10_B$ are prohibited)**

### Start

- Start the TAUJ0.
- Set the SCTLR.ADEN bit to 1.

After the operation starts, ports are checked at the interval set in TAUJ0. The operation continues regardless of whether the mode is RUN mode or power save mode. If the HS IntOSC is stopped in stand-by mode, it can only resume operation while the sequencer is running.

To detect whether the analog input value differs from the expected value, use the A/D error interrupt request (INTADCA0ERR) of the A/D converter.

In addition, if the analog input value is different from the expected value, the wake-up factor WUTRG1 occurs.

For details on the A/D error interrupt request (INTADCA0ERR), see **Section 31.4.13, A/D Error Interrupt Request**[*1]. The following figures show an example of the operation in analog input mode.

**Note 1.** In **Section 31 A/D Converter (ADCA)**, the name of the A/D error interrupt request is described as “INT_ADE”.

### Stop

To stop the LPS operation in Analog Input Mode (by changing the SCTLR.ADEN bit setting from 1 to 0), follow the procedure shown below. Note that the P0_1 pin is used as APO.

1. Set the port register to specify low level output on the pin (P0.P0_1 = 0).
2. Change the setting for the P0_1 pin from the alternative port mode to the port mode (PMC0.PMC0_1 = 0).
3. Set SCTLR.ADEN = 0.

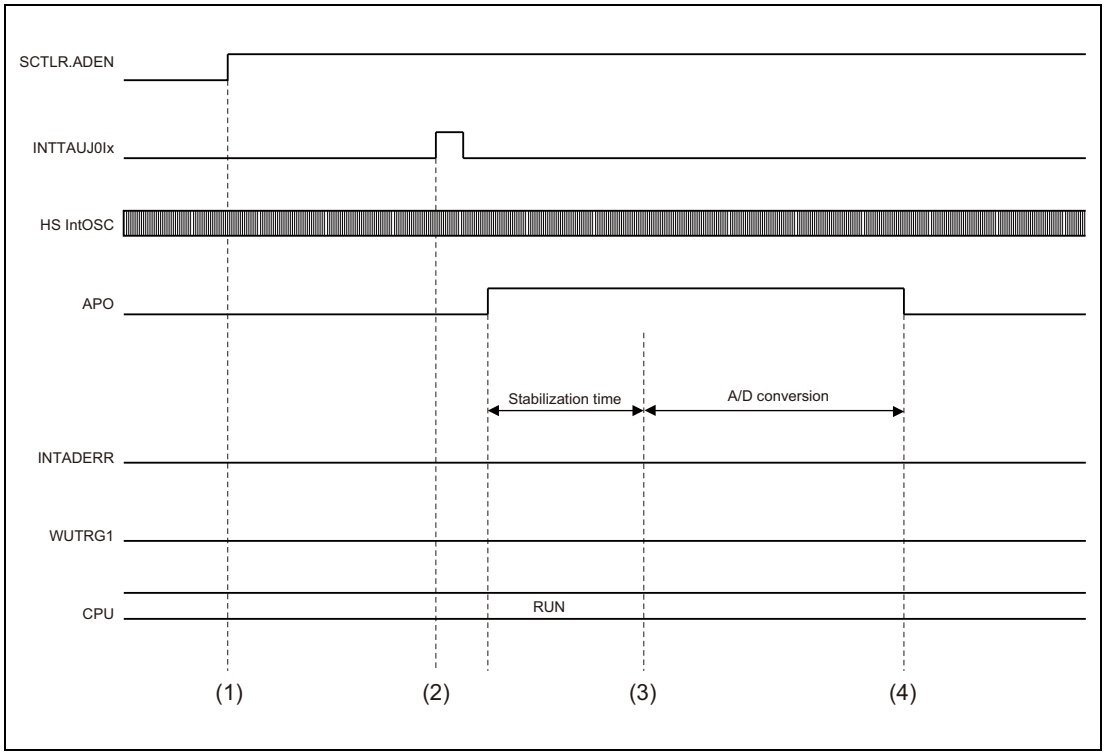



**Figure 15.9 Operation of Analog Input Mode when the Conversion Result is within the Expected Range (RUN Mode)**

(1) Set the conversion trigger, scan group, and expected range of the A/D converter by software. Then, set the SCTLR.ADEN bit to 1 to enable the analog input mode of the LPS.

(2) When the INTTAUJ0Ix interrupt specified by the SCTLR.TJIS bit is generated, the LPS outputs the high level from the APO pin at the same time it enables the A/D converter, and waits for the time specified by CNTVAL.CNT1n to secure the stabilization of the external analog signal source. Set the stabilization time not less than 1 µs.

(3) After the completion of the signal source stabilization, the LPS triggers the start of conversion to the A/D converter and then the A/D conversion of ADCA0Im (m = 0 to 15), set in the A/D converter scan group, is started.

(4) When the INTADCA0ERR interrupt is not generated as a result of A/D conversion, the LPS halts the A/D converter and resets the APO pin.

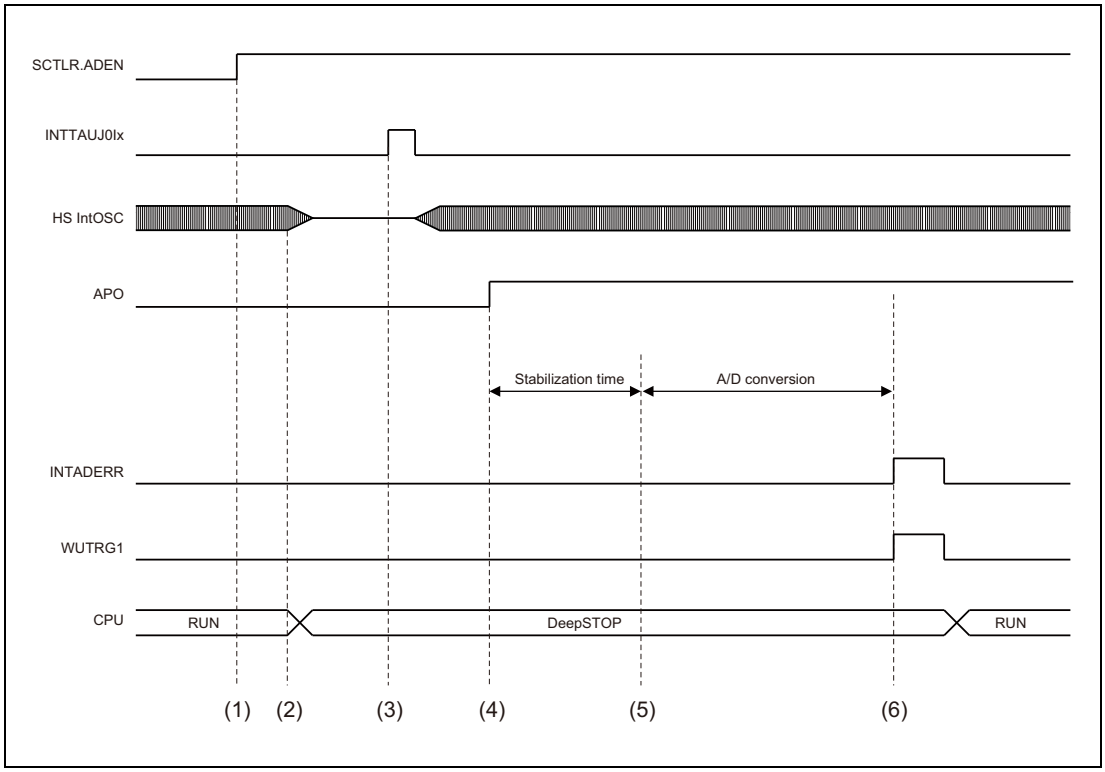



**Figure 15.10 Operation of Analog Input Mode when the Conversion Result is not within the Expected Range (DeepSTOP Mode)**

(1) Set the conversion trigger, scan group, and expected range of the A/D converter by software. Then, set the SCTLR.ADEN bit to 1 to enable the analog input mode of the LPS.

(2) Set the STBC0PSC.STBC0DISTRG bit to 1 by software to transition to the DeepSTOP mode.

(3) When the INTTAUJ0Ix interrupt specified by the SCTLR.TJIS bit is generated, the LPS enables the HS IntOSC to start the oscillation.

(4) After the completion of the HS IntOSC stabilization, the LPS outputs the high level from the APO pin at the same time it enables the A/D converter, and waits for the time specified by CNTVAL.CNT1n to secure the stabilization of the external analog signal source.

(5) After the completion of the signal source stabilization, the LPS triggers the start of conversion to the A/D converter and then the A/D conversion of ADCA0Im (m = 0 to 15), set in the A/D converter scan group, is started.

(6) When the INTADCA0ERR interrupt is generated as a result of A/D conversion, the wake-up factor WUTRG1 is generated and the CPU returns to RUN mode. The APO pin is driven high until the upper limit/lower limit error flag of the A/D converter is cleared to 0 by software.

**NOTE**

WUTRG1 is generated when the first ADCA0 SG interrupt occurred after ADCAnULER.UE or ADCAnULER.LE is set.

### 15.5.1 Analog Port Error Interrupt

A level sensitive interrupt indicating a data comparison mismatch is generated. This interrupt is generated not only in the stand-by mode but also in RUN mode. The set and clear conditions are shown below.

**Table 15.32 Analog Port Error Interrupt**

| Unit Interrupt Signal | Set Condition | Clear Condition |
|---|---|---|
| INTAPE | When WUTRG1 is set to 1 by hardware | When both ADCAnULER.UE and ADCAnULER.LE are cleared to 0 by software |

# Section 16 Clocked Serial Interface G (CSIG)

This section contains a generic description of the Clocked Serial Interface G (CSIG).

The first part in this section describes the features specific to RH850/F1K, such as the number of units, register base addresses, etc. The remainder of the section describes the functions and registers of the CSIG.

## 16.1 Features of RH850/F1K CSIG

### 16.1.1 Number of Units

This microcontroller has the following number of CSIG units.

Each CSIG unit has one channel interface. “Number of channels” is used with the same meaning as “number of units” in this section.

**Table 16.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of Units | 1 | 2 | 2 |
| Name | CSIGn (n = 0) | CSIGn (n = 0, 1) | CSIGn (n = 0, 1) |

**Table 16.2 Index**

| Index | Description |
|---|---|
| n | Throughout this section, the individual CSIG units are identified by the index “n” (n = 0, 1): for example, CSIGnCTL0 is the CSIGn control register 0. |

### 16.1.2 Register Base Address

CSIG base addresses are listed in the following table.

CSIG register addresses are given as offsets from the base addresses.

**Table 16.3 Register Base Addresses**

| Base Address Name | Base Address |
|---|---|
| <CSIG0_base> | FFD8 $8000_H$ |
| <CSIG1_base> | FFD8 $A000_H$ |

### 16.1.3 Clock Supply

The CSIG clock supply is shown in the following table.

**Table 16.4 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| CSIGn | PCLK | CKSCLK_ICSI | Communication clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CKSCLK_ICSI | |

### 16.1.4 Interrupt Request

CSIG interrupt requests are listed in the following table.

**Table 16.5 Interrupt Requests**

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **CSIG0** | | | |
| INTCSIGTIC | Communication status interrupt | 27,118 | 8 |
| INTCSIGTIR | Receive status interrupt | 28,119 | 9 |
| INTCSIGTIRE | Communication error interrupt | 57 | — |
| **CSIG1** | | | |
| INTCSIGTIC | Communication status interrupt | 223 | 109 |
| INTCSIGTIR | Receive status interrupt | 224 | 110 |
| INTCSIGTIRE | Communication error interrupt | 225 | — |

### 16.1.5 Reset Sources

CSIG reset sources are listed in the following table. CSIG is initialized by these reset sources.

**Table 16.6 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| CSIGn | All reset sources (ISORES) |

### 16.1.6 External Input/Output Signals

External input/output signals of CSIG are listed below.

**Table 16.7 External Input/Output Signals**

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **CSIG0** | | |
| CSIGTSCK | Serial clock signal | CSIG0SC |
| CSIGTSI | Serial data input signal | CSIG0SI |
| CSIGTSO | Serial data output signal | CSIG0SO |
| $\overline{\text{CSIGTSSI}}$ | Slave select input signal | $\overline{\text{CSIG0SSI}}$ |
| CSIGTRYI | Ready / busy input signal | CSIG0RYI |
| CSIGTRYO | Ready / busy output signal | CSIG0RYO |
| **CSIG1** | | |
| CSIGTSCK | Serial clock signal | CSIG1SC |
| CSIGTSI | Serial data input signal | CSIG1SI |
| CSIGTSO | Serial data output signal | CSIG1SO |
| $\overline{\text{CSIGTSSI}}$ | Slave select input signal | $\overline{\text{CSIG1SSI}}$ |
| CSIGTRYI | Ready / busy input signal | CSIG1RYI |
| CSIGTRYO | Ready / busy output signal | CSIG1RYO |

**Note:** For the port pins that are used as CSIGnSO and CSIGnSC, set the output driving ability to high (PDSCn_m = 1).

### 16.1.7 Data Consistency Check

The port and the alternative function for data consistency check of CSIGnSO (CSIGTSO) output are shown in the following table. See **Section 16.5.10, Error Detection** for details on data consistency checking.

**Table 16.8 Port Pins for Data Consistency Checking**

| Unit Signal Name | Port Pin Name | Alternative Function |
|---|---|---|
| **CSIG0** | | |
| CSIGTSO | P0_13 | ALT_OUT4 |
| | P10_6 | ALT_OUT2 |
| **CSIG1** | | |
| CSIGTSO | P11_9 | ALT_OUT1 |

## 16.2 Overview

### 16.2.1 Functional Overview

- Three-wire serial synchronous data transfer
- Master mode or slave mode is selectable.
- Slave select input signal ($\overline{\text{CSIGTSSI}}$) is available.
- Built-in baud rate generator
- Transfer clock frequency is adjustable in master mode, whereas it is determined by the input clock in slave mode.
- Maximum transfer clock frequency:
  - In master mode: 10.0 MHz (however, it must be equal to or lower than PCLK/4)
  - In slave mode: 5.0 MHz (however, it must be equal to or lower than PCLK/16)
- Clock phases and data phases are selectable.
- Data transfer with MSB first or LSB first is selectable.
- Transfer data length is selectable from 7 to 16 bits in 1-bit units
- Built-in EDL (extended data length) function for transferring more than 16 bits of data.
- Three selectable transfer modes:
  - transmit-only mode
  - receive-only mode
  - transmit/receive mode
- Built-in handshake function
- Built-in error detection (data consistency check, parity, overrun)
- Three different interrupt request signals (INTCSIGTIC, INTCSIGTIR, INTCSIGTIRE)
- Built-in LBM (Loop Back Mode) function for self-test

### 16.2.2 Functional Overview Description

The CSIG uses three signals for communication:

- Transmission clock CSIGTSCK (output in master mode, input in slave mode)
- Serial data output signal CSIGTSO
- Serial data input signal CSIGTSI

The CSIGnCTL2 register is used to select whether the CSIG should be operated in master mode or slave mode.

Additional signals can be used for external control and monitoring:

- CSIGTSSI: Slave select input signal
- CSIGTRYO: Ready/busy output signal (handshake signal)
- CSIGTRYI: Ready/busy input signal (handshake signal)

Data transmission is bit-wise and serial and synchronous to the transmission clock.

The following table shows the most important registers for setting up CSIG.

**Table 16.9 Main Registers of CSIG**

| Register | Function |
|---|---|
| CSIGnCTL0 | Provides and stops operating clock and enables/disables data transmission and data reception. |
| CSIGnCTL1 | Controls options such as interrupt timing, extended data length, data consistency check, loop-back mode, handshake, etc. |
| CSIGnCTL2 | Selects master or slave mode, and the transfer clock frequency of the built-in baud rate generator (BRG) in master mode. |
| CSIGnCFG0 | Configures the communication protocol. |

## 16.2.3 Block Diagram

The following block diagram shows the main components of the CSIG.

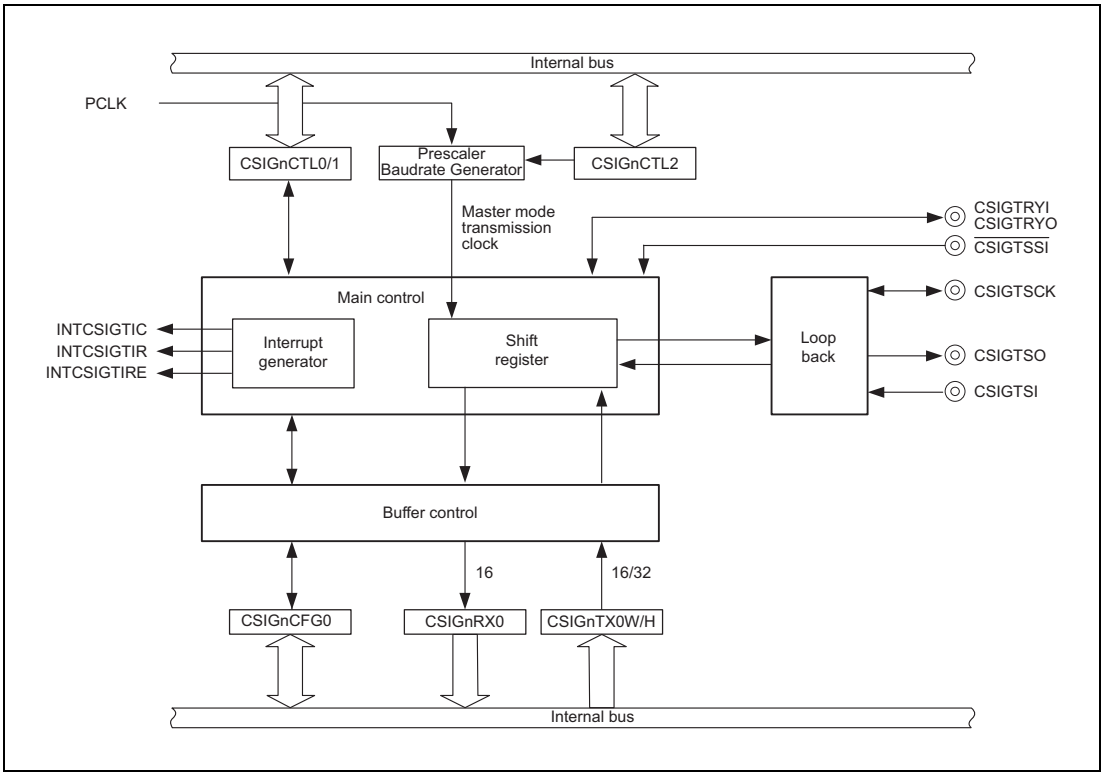



**Figure 16.1 CSIG Block Diagram**

In master mode, the transmission clock CSIGTSCK is generated by the built-in baud rate generator (BRG). In slave mode, the transmission clock is supplied by an external source.

# 16.3 Registers

## 16.3.1 List of Registers

CSIG registers are listed in the following table.

For details on <CSIGn_base>, see **Section 16.1.2, Register Base Address**.

**Table 16.10 List of Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| CSIGn | CSIGn control register 0 | CSIGnCTL0 | <CSIGn_base> + $0000_H$ |
| | CSIGn control register 1 | CSIGnCTL1 | <CSIGn_base> + $0010_H$ |
| | CSIGn control register 2 | CSIGnCTL2 | <CSIGn_base> + $0014_H$ |
| | CSIGn status register 0 | CSIGnSTR0 | <CSIGn_base> + $0004_H$ |
| | CSIGn status clear register 0 | CSIGnSTCR0 | <CSIGn_base> + $0008_H$ |
| | CSIGn Receive-only mode control register 0 | CSIGnBCTL0 | <CSIGn_base> + $1000_H$ |
| | CSIGn configuration register 0 | CSIGnCFG0 | <CSIGn_base> + $1010_H$ |
| | CSIGn transmission register 0 for word access | CSIGnTX0W | <CSIGn_base> + $1004_H$ |
| | CSIGn transmission register 0 for half word access | CSIGnTX0H | <CSIGn_base> + $1008_H$ |
| | CSIGn reception register 0 | CSIGnRX0 | <CSIGn_base> + $100C_H$ |
| | CSIGn emulation register | CSIGnEMU | <CSIGn_base> + $0018_H$ |

## 16.3.2 CSIGnCTL0 — CSIGn Control Register 0

This register controls the operation clock, and enables or disables transmission/reception.

**Access:** This register can be read or written in 8-bit units.

**Address:** <CSIGn_base> + $0000_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CSIGnPWR | CSIGnTXE | CSIGnRXE | — | — | — | — | CSIGnMBS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R | R | R | R | R/W |

**Table 16.11 CSIGnCTL0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | CSIGnPWR | Controls operation clock.<br>0: Stops operation clock.<br>1: Supplies operation clock.<br>Clearing CSIGnPWR to 0 resets the internal circuits, stops operation, and sets the CSIG to standby state. Clock supply to internal circuits stops.<br>If CSIGnPWR is cleared during communication, ongoing communication is aborted. In this case, communication setting must be started from the beginning. |
| 6 | CSIGnTXE | Enables/disables transmission.<br>0: Transmission disabled<br>1: Transmission enabled |
| 5 | CSIGnRXE | Enables/disables reception.<br>0: Reception disabled<br>1: Reception enabled |
| 4 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | CSIGnMBS | This bit must be set to 1 (the value after reset is 0). |

**CAUTION**

When configuring this register, see **Table 16.24, List of Cautions when Configuring the Registers**.

### 16.3.3 CSIGnCTL1 — CSIGn Control Register 1

This register specifies the interrupt timing and the interrupt delay mode. It also enables/disables extended data length control, data consistency check, loop-back mode, handshake function, and slave select function.

**Access:** This register can be read or written in 32-bit units.

**Address:** <CSIGn_base> + $0010_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CSIGn CKR | CSIGn SLIT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CSIGn EDLE | — | CSIGn DCS | — | CSIGn LBM | CSIGn SIT | CSIGn HSE | CSIGn SSE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R | R/W | R | R/W | R/W | R/W | R/W |

**Table 16.12 CSIGnCTL1 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 18 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 17 | CSIGnCKR | CSIGTSCK clock inversion function<br>0: Default level of CSIGTSCK is high.<br>1: Default level of CSIGTSCK is low.<br>The CSIGnCKR bit is used in combination with the CSIGnCFG0.CSIGnDAP bit.<br>For details, see **Section 16.3.8, CSIGnCFG0 — CSIGn Configuration Register 0**. |
| 16 | CSIGnSLIT | Selects the timing of interrupt INTCSIGTIC.<br>0: Normal interrupt timing (interrupt is generated after the transfer)<br>1: Interrupt generation when CSIGnTX0W/H is empty and available for storing the next data.<br>For details, see **16.4.2, INTCSIGTIC (Communication Status Interrupt)** |
| 15 to 8 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 7 | CSIGnEDLE | Enables/disables extended data length (EDL) mode.<br>0: Extended data length mode disabled<br>1: Extended data length mode enabled<br>For details, see **16.5.5.2, Data Length Selection with Extended Data Length**. |
| 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 | CSIGnDCS | Enables/disables data consistency check.<br>0: Data consistency check disabled<br>1: Data consistency check enabled<br>For details, see **16.5.10.1, Data Consistency Check**. |
| 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 | CSIGnLBM | Controls loop-back mode (LBM).<br>0: Loop-back mode deactivated<br>1: Loop-back mode activated<br>Loop-back mode can be set only in master mode. Set this bit to 0 in slave mode.<br>For details, see **Section 16.5.9, Loop-Back Mode**. |

**Table 16.12 CSIGnCTL1 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 2 | CSIGnSIT | Selects interrupt delay mode.<br>0: No delay<br>1: Half clock delay for all interrupts<br>This bit is only valid in master mode. In slave mode, no delay is generated.<br>For details, see **Section 16.4.1, Interrupt Delay**. |
| 1 | CSIGnHSE | Enables/disables handshake function.<br>0: Handshake function disabled<br>1: Handshake function enabled<br>For details, see **Section 16.5.8, Handshake Function**. |
| 0 | CSIGnSSE | Enables/disables slave select function.<br>0: Input signal $\overline{\text{CSIGTSSI}}$ disabled.<br>1: Input signal $\overline{\text{CSIGTSSI}}$ enabled.<br>If the slave select function is not used, this bit must be set to 0 (see also **Section 16.5.2, Master/Slave Connections**). |

Details about CSIGnCTL1.CSIGnSSE:

**Table 16.13 Operation of the Slave Select Function during Reception**

| CSIGnCTL0. CSIGnRXE | CSIGnCTL1. CSIGnSSE | $\overline{\text{CSIGTSSI}}$ | Receive Operation |
|---|---|---|---|
| 0 | — | — | Reception disabled |
| 1 | 0 | — | Possible |
| 1 | 1 | 0 | Possible |
| 1 | 1 | 1 | Disabled |

**Table 16.14 Operation of the Slave Select Function during Transmission**

| CSIGnCTL0. CSIGnTXE | CSIGnCTL1. CSIGnSSE | $\overline{\text{CSIGTSSI}}$ | Transmit Operation |
|---|---|---|---|
| 0 | — | — | Transmission disabled |
| 1 | 0 | — | Possible |
| 1 | 1 | 0 | Possible |
| 1 | 1 | 1 | Disabled |

**CAUTION**

When configuring this register, see **Table 16.24, List of Cautions when Configuring the Registers**.

## 16.3.4 CSIGnCTL2 — CSIGn Control Register 2

This register selects the communication clock.

**Access:** This register can be read or written in 16-bit units.

**Address:** <CSIGn_base> + $0014_H$

**Value after reset:** $E000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIGnPRS[2:0] | | | — | CSIGnBRS[11:0] | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 16.15 CSIGnCTL2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 13 | CSIGnPRS [2:0] | Selects the value of the prescaler. |
| 12 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 11 to 0 | CSIGnBRS [11:0] | Selects the transfer clock frequency.<br>Settings of the CSIGnBRS[11:0] bits are valid only in master mode. They are ignored in slave mode. |

| CSIGn PRS2 | CSIGn PRS1 | CSIGn PRS0 | Prescaler Output (PRSOUT) |
|---|---|---|---|
| 0 | 0 | 0 | PCLK (master mode) |
| 0 | 0 | 1 | PCLK / 2 (master mode) |
| 0 | 1 | 0 | PCLK / 4 (master mode) |
| 0 | 1 | 1 | PCLK / 8 (master mode) |
| 1 | 0 | 0 | PCLK / 16 (master mode) |
| 1 | 0 | 1 | PCLK / 32 (master mode) |
| 1 | 1 | 0 | PCLK / 64 (master mode) |
| 1 | 1 | 1 | External clock via CSIGTSCK (slave mode) |

| CSIGnBRS [11:0] | Transfer Clock Frequency of CSIGTSCK |
|---|---|
| 0 | BRG is stopped |
| 1 | PCLK / ($2^{\alpha}$ × 1 × 2) |
| 2 | PCLK / ($2^{\alpha}$ × 2 × 2) |
| 3 | PCLK / ($2^{\alpha}$ × 3 × 2) |
| 4 | PCLK / ($2^{\alpha}$ × 4 × 2) |
| ... | ... |
| 4095 | PCLK / ($2^{\alpha}$ × 4095 × 2) |

**Note:** α = 0 to 6 (value set by CSIGnPRS[2:0])

**CAUTION**

When configuring this register, see **Table 16.24, List of Cautions when Configuring the Registers**.

## 16.3.5 CSIGnSTR0 — CSIGn Status Register 0

This register indicates the status of the CSIG.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <CSIGn_base> + $0004_H$

**Value after reset:** $0000\ 0010_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CSIGn TSF | — | — | — | CSIGn DCE | — | CSIGn PE | CSIGn OVE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 16.16 CSIGnSTR0 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned. |
| 7 | CSIGnTSF | Transfer Status Flag<br>0: Idle state<br>1: Communication is in progress or being prepared<br>The timing to set or clear this bit is as follows:<br><br>**Master Mode / Timing to Set / Timing to Clear**<br>Tx-only mode: Writing to transmission register: Within a half clock cycle from the last serial clock edge<br>Tx/Rx mode: Writing to transmission register: Within a half clock cycle from the last serial clock edge<br>Rx-only mode: Reading from reception register: Within a half clock cycle from the last serial clock edge<br><br>**Slave Mode / Timing to Set / Timing to Clear**<br>Tx-only mode: Writing to transmission register: Within a half clock cycle from the last serial clock edge<br>Tx/Rx mode: Writing to transmission register: Within a half clock cycle from the last serial clock edge<br>Rx-only mode: CSIGTSCK input: Within a half clock cycle from the last serial clock edge |
| 6 to 4 | Reserved | When read, the value after reset is returned. |
| 3 | CSIGnDCE | Data Consistency Check Error Flag<br>0: No data consistency check error detected<br>1: Data consistency check error detected<br>This bit is cleared by writing 1 to CSIGnSTCR0.CSIGnDCEC.<br>When setting to 1 due to data consistency check error detection and clearing to 0 by CSIGnSTCR0.CSIGnDCEC occur simultaneously, setting to 1 due to data consistency check error detection takes precedence.<br>This bit is initialized when CSIGnCTL0.CSIGnPWR changes from 0 to 1, or from 1 to 0. |
| 2 | Reserved | When read, the value after reset is returned. |

**Table 16.16 CSIGnSTR0 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 1 | CSIGnPE | Parity Error Flag<br>0: No parity error detected<br>1: Parity error detected<br>This bit is cleared by writing 1 to CSIGnSTCR0.CSIGnPEC.<br>When setting to 1 due to parity error detection and clearing to 0 by writing to CSIGnSTCR0.CSIGnPEC occur simultaneously, setting to 1 due to parity error detection takes precedence.<br>This bit is initialized when CSIGnCTL0.CSIGnPWR changes from 0 to 1, or from 1 to 0. |
| 0 | CSIGnOVE | Overrun Error Flag<br>0: No overrun error detected<br>1: Overrun error detected<br>This bit is cleared by writing 1 to CSIGnSTCR0.CSIGnOVEC.<br>When setting to 1 due to overrun error detection and clearing to 0 by writing to CSIGnSTCR0.CSIGnOVEC occur simultaneously, setting to 1 due to overrun error detection takes precedence.<br>This bit is initialized by the value of CSIGnCTL0.CSIGnPWR changes from 0 to 1, or from 1 to 0. |

**CAUTION**

When configuring this register, see **Table 16.24, List of Cautions when Configuring the Registers**.

### 16.3.6 CSIGnSTCR0 — CSIGn Status Clear Register 0

This register clears the status flags of the CSIGnSTR0 status register.

**Access:** This register can be read or written in 16-bit units.
When read, the value $0000_H$ is always returned.

**Address:** <CSIGn_base> + $0008_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | CSIGn DCEC | — | CSIGn PEC | CSIGn OVEC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R/W | R/W |

**Table 16.17 CSIGnSTCR0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 | CSIGnDCEC | Controls the data consistency check error flag clear command.<br>0: No operation. Read value is always 0.<br>1: Clears the data consistency check error flag (CSIGnSTR0.CSIGnDCE). |
| 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | CSIGnPEC | Controls the parity error flag clear command.<br>0: No operation. Read value is always 0.<br>1: Clears the parity error flag (CSIGnSTR0.CSIGnPE). |
| 0 | CSIGnOVEC | Controls the overrun error flag clear command.<br>0: No operation. Read value is always 0.<br>1: Clears the overrun error flag (CSIGnSTR0.CSIGnOVE). |

## 16.3.7 CSIGnBCTL0 — CSIGn Rx-Only Mode Control Register 0

This register enables/disables the data transfer in Rx-only mode.

**Access:** This register can be read or written in 8-bit or 1-bit units.

**Address:** <CSIGn_base> + $1000_H$

**Value after reset:** $01_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CSIGnSCE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 16.18 CSIGnBCTL0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | CSIGnSCE | Disables/enables the start of the next data reception by reading CSIGnRX0.<br>0: Next reception disabled<br>1: Next reception enabled<br>For details, see **Section 16.5.4.2, Receive-Only Mode**. |

**CAUTION**

When configuring this register, see **Table 16.24, List of Cautions when Configuring the Registers**.

## 16.3.8 CSIGnCFG0 — CSIGn Configuration Register 0

This register configures the communication protocols such as data length, parity, transfer direction, clock phase, and data phase.

**Access:** This register can be read or written in 32-bit units.

**Address:** <CSIGn_base> + $1010_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | CSIGnPS[1:0] | | CSIGnDLS[3:0] | | | | — | — | — | — | — | CSIGn DIR | — | CSIGn DAP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R/W | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 16.19 CSIGnCFG0 Register Contents (1/2)**

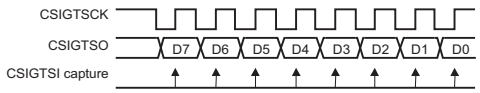
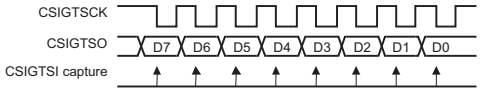
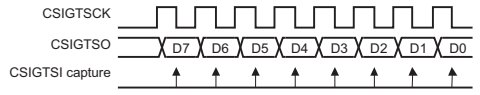
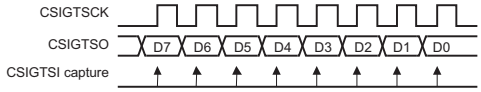
| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 29, 28 | CSIGnPS[1:0] | Specifies parity. |

| CSIGn PS1 | CSIGn PS0 | Transmission | Reception |
|---|---|---|---|
| 0 | 0 | Does not transmit any parity bit. | Does not wait for reception of the parity bit. |
| 0 | 1 | Adds a parity bit fixed to 0. | Waits for reception of the parity bit but does not evaluate it. |
| 1 | 0 | Adds the odd parity bit. | Waits for the odd parity bit. |
| 1 | 1 | Adds the even parity bit. | Waits for the even parity bit. |

| | | |
|---|---|---|
| 27 to 24 | CSIGnDLS [3:0] | Specifies data length.<br>0: Data length is 16 bits<br>1: Data length is 1 bit<br>2: Data length is 2 bits<br>...<br>15: Data length is 15 bits<br>**CAUTION**<br>Do not set bits CSIGnCFG0.CSIGnDLS[3:0] to values 1 to 6 when the extended data length function is disabled with bit CSIGnCTL1.CSIGnEDLE set to 0.<br>Transmitting two consecutive data with a data length of less than 7 bits is prohibited. |
| 23 to 19 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 18 | CSIGnDIR | Selects the serial data direction.<br>0: Data is transmitted/received with MSB first<br>1: Data is transmitted/received with LSB first |
| 17 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

**Table 16.19 CSIGnCFG0 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 16 | CSIGnDAP | Data Phase Selection<br>Used in conjunction with CSIGnCTL1.CSIGnCKR bit to select the data phase.<br>Refer to the following table for the selectable clock phases and data phases. |
| 15 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

| CSIGnCTL1.CSIGnCKR | CSIGnDAP | Clock and Data Phase Selection |
|---|---|---|
| 0 | 0 |  |
| 0 | 1 |  |
| 1 | 0 |  |
| 1 | 1 |  |

**CAUTION**

When configuring this register, see **Table 16.24, List of Cautions when Configuring the Registers**.

## 16.3.9 CSIGnTX0W — CSIGn Transmission Register 0 for Word Access

This register stores the transmission data and specifies the extended data length.

**Access:** This register can be read or written in 32-bit units.

**Address:** <CSIGn_base> + $1004_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | CSIGnEDL | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIGnTX[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 16.20 CSIGnTX0W Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 29 | CSIGnEDL | Specifies the extended data length.<br>0: Normal operation<br>1: Extended data length enabled<br>The associated data is transmitted as 16-bit data.<br>This bit can only be set if CSIGnCTL1.CSIGnEDLE = 1. |
| 28 to 16 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 15 to 0 | CSIGnTX[15:0] | Data to be transmitted |

**CAUTION**

When configuring this register, see **Table 16.24, List of Cautions when Configuring the Registers**.

## 16.3.10 CSIGnTX0H — CSIGn Transmission Register 0 for Half Word Access

This register stores the transmission data. This register is the same as bits 15 to 0 of CSIGnTX0W register.
The settings specified by the upper 16 bits of CSIGnTX0W are applied to the transmission.

**Access:** This register can be read or written in 16-bit units.

**Address:** <CSIGn_base> + $1008_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIGnTX[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 16.21 CSIGnTX0H Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | CSIGnTX[15:0] | Data to be transmitted |

**CAUTION**

When configuring this register, see **Table 16.24, List of Cautions when Configuring the Registers**.

## 16.3.11 CSIGnRX0 — CSIGn Reception Register 0

This register stores the received data.

**Access:** This register is a read-only register that can be read in 16-bit units.

**Address:** <CSIGn_base> + $100C_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIGnRX[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 16.22 CSIGnRX0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | CSIGnRX [15:0] | Received Data<br>These bits are initialized when CSIGnCTL0.CSIGnPWR changes from 0 to 1 or from 1 to 0.<br>When reading, the values of these bits must be read at least 1 clock before the generation of CSIGTIR interrupt. |

**CAUTION**

When configuring this register, see **Table 16.24, List of Cautions when Configuring the Registers**.

## 16.3.12 CSIGnEMU — CSIGn Emulation Register

This register controls operation by SVSTOP.

**Access:** This register can be read or written in 8-bit or 1-bit units.
Write to this register when EPC.SVSTOP = 0.

**Address:** <CSIGn_base> + $0018_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CSIGnSVSDIS | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R |

**Table 16.23 CSIGnEMU Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | CSIGnSVSDIS | Selects whether to continue or stop transmit/receive operation during debugging.<br>• When the EPC.SVSTOP bit is set to 0:<br>Continues transmit/receive operation regardless of the setting of this bit.<br>• When the EPC.SVSTOP bit is set to 1:<br>0: Stops transmit/receive operation.<br>1: Continues transmit/receive operation. |
| 6 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

**CAUTION**

When configuring this register, see **Table 16.24, List of Cautions when Configuring the Registers**.

### 16.3.13 List of Cautions

Table 16.24 List of Cautions when Configuring the Registers

| Register Name | Bit Name | Cautions |
|---|---|---|
| CSIGnCTL0 | CSIGnPWR | If this bit is cleared during communication, ongoing communication is aborted. After the communication is aborted, it is necessary to restart the communication. |
| CSIGnCTL0 | CSIGnTXE<br>CSIGnRXE | Do not modify any of these bits while CSIGnCTL0.CSIGnPWR = 0. (These bits can be modified simultaneously with the CSIGnCTL0.CSIGnPWR bit.)<br>Do not modify these bits while CSIGnSTR0.CSIGnTSF = 1, because the specified operation is not guaranteed if ongoing communication is aborted. |
| CSIGnCTL0 | CSIGnMBS | When writing, be sure to set to this bit to 1. (The value after reset is “0”.)<br>This bit must be modified simultaneously with CSIGnCTL0.CSIGnPWR bit. |
| CSIGnCTL1 | CSIGnCKR | Modification of this bit is only permitted while CSIGnCTL0.CSIGnPWR = 0. |
| CSIGnCTL1 | CSIGnSLIT<br>CSIGnEDLE<br>CSIGnDCS<br>CSIGnHSE | Modification of these bits is only permitted while CSIGnCTL0.CSIGnPWR = 0. |
| CSIGnCTL1 | CSIGnLBM | Modification of this bit is only permitted while CSIGnCTL0.CSIGnPWR = 0.<br>Setting of this bit is prohibited in slave mode. |
| CSIGnCTL1 | CSIGnSSE | Modification of this bit is only permitted while CSIGnCTL0.CSIGnPWR = 0.<br>Setting this bit to 1 is prohibited in master mode. |
| CSIGnCTL1 | CSIGnSIT | Modification of this bit is only permitted while CSIGnCTL0.CSIGnPWR = 0.<br>This bit is only valid in master mode. In slave mode, no delay is generated. |
| CSIGnCTL2 | CSIGnPRS[2:0]<br>CSIGnBRS[11:0] | Modification of these bits is only permitted while CSIGnCTL0.CSIGnPWR = 0.<br>Setting of the maximum transfer clock frequency is as follows.<br>• Master mode: 10.0 MHz (however, it must be equal to or lower than PCLK/4)<br>• Slave mode: 5.0 MHz (however, it must be equal to or lower than PCLK/16) |
| CSIGnSTR0 | CSIGnTSF | Writing to this bit is prohibited, and only reading is permitted. |
| CSIGnSTR0 | CSIGnDCE<br>CSIGnPE<br>CSIGnOVE | Writing to these bits is prohibited, and only reading is permitted.<br>These bits are initialized when CSIGnCTL0.CSIGnPWR = 0 → 1 or CSIGnCTL0.CSIGnPWR = 1 → 0. |
| CSIGnBCTL0 | CSIGnSCE | Write to this bit before CSIGnRX0 is read.<br>Fix the CSIGnSCE bit to 0 when the transfer mode is transmit mode or transmit/receive mode. |
| CSIGnCFG0 | CSIGnPS[1:0]<br>CSIGnDLS[3:0]<br>CSIGnDIR<br>CSIGnDAP | Modification of these bits is only permitted while CSIGnCTL0.CSIGnPWR = 0. |
| CSIGnTX0W | CSIGnEDL | This bit is valid only when CSIGnCTL1.CSIGnEDLE = 1. |
| CSIGnTX0W<br>CSIGnTX0H | | Write access to these bits are prohibited when CSIGnCTL0.CSIGnTXE = CSIGnCTL0.CSIGnRXE = 0. |
| CSIGnRX0 | | These bits are initialized when CSIGnCTL0.CSIGnPWR = 0 → 1 or CSIGnCTL0.CSIGnPWR = 1→ 0.<br>Read access to this bit is prohibited when CSIGnCTL0.CSIGnTXE = CSIGnCTL0.CSIGnRXE = 0. |
| CSIGnEMU | CSIGnSVSDIS | Modification of this bit is prohibited while SVSTOP = 1. |

## 16.4 Interrupt Sources

CSIG can generate the following interrupts:

- INTCSIGTIC (communication status interrupt)
- INTCSIGTIR (reception status interrupt)
- INTCSIGTIRE (communication error interrupt)

### 16.4.1 Interrupt Delay

In master mode, all interrupts generated by the master can be delayed by half cycle of the transmission clock CSIGTSCK. This function is not available in slave mode.

The delay is specified by setting CSIGnCTL1.CSIGnSIT = 1. (The setting of the CSIGnSIT bit is invalid in slave mode.)

The following example illustrates the interrupt delay function, assuming a setting of CSIGnCTL1.CSIGnSIT = 1 (interrupt delay enabled), CSIGnCTL1.CSIGnCKR = 0, CSIGnCFG0.CSIGnDAP = 0 (normal clock and data phase), and CSIGnCFG0.CSIGnDLS[3:0] = $1000_B$ (data length 8 bits).

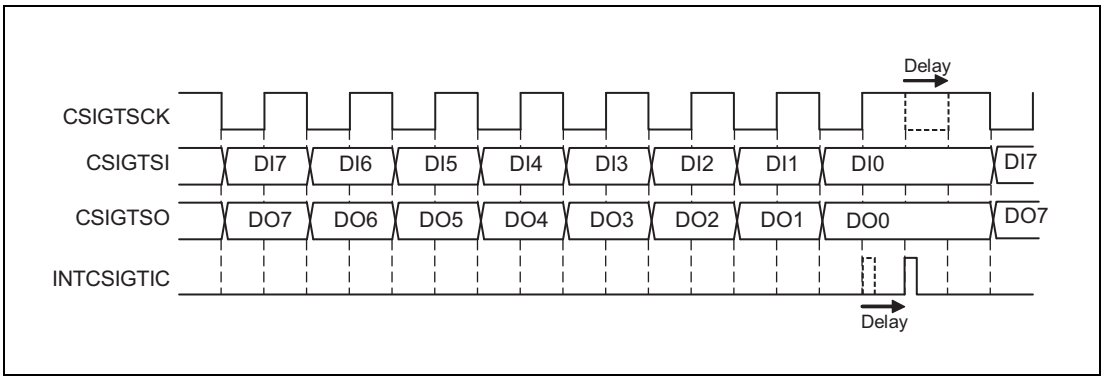



**Figure 16.2 Interrupt Delay Function (CSIGnCTL1.CSIGnSIT = 1)**

### 16.4.2 INTCSIGTIC (Communication Status Interrupt)

This interrupt is normally generated after every data transfer. It can be used to trigger a DMA for writing new transmission data to register CSIGnTX0W or CSIGnTX0H.

The following example assumes master mode and a setting of CSIGnCTL1.CSIGnSIT = 0 (no interrupt delay), CSIGnCTL1.CSIGnCKR = 0, CSIGnCFG0.CSIGnDAP = 0 (normal clock and data phase), CSIGnCFG0.CSIGnDLS[3:0] = $1000_B$ (data length 8 bits), and CSIGnCTL1.CSIGnSLIT = 0 (normal interrupt timing).

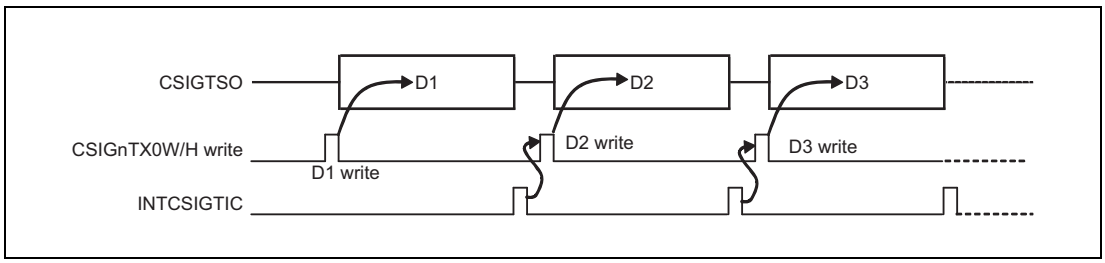



**Figure 16.3 Generation of INTCSIGTIC after Communication (CSIGnCTL1.CSIGnSLIT = 0)**

However, INTCSIGTIC can also be set up to occur when the CSIGnTX0W/H register is empty and available for receiving the next data. This is specified by setting CSIGnCTL1.CSIGnSLIT = 1.

This mode allows more efficient data transfers.

The effect is illustrated in the figure below.

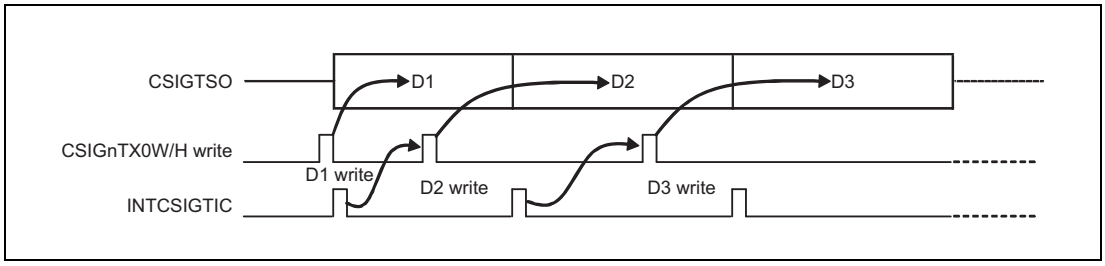



**Figure 16.4 Generation of INTCSIGTIC at the Beginning of Communication**

### 16.4.3 INTCSIGTIR (Reception Status Interrupt)

This interrupt is generated in receive-only and transmit/receive mode after data has been received and is available in the reception register. It can be used to trigger a DMA for reading the received data from CSIGnRX0 register.

The following example assumes master mode and a setting of CSIGnCTL1.CSIGnSIT = 0 (no interrupt delay), CSIGnCTL1.CSIGnCKR = 0, CSIGnCFG0.CSIGnDAP = 0 (normal clock and data phase), and CSIGnCFG0.CSIGnDLS[3:0] = $1000_B$ (data length 8 bits).

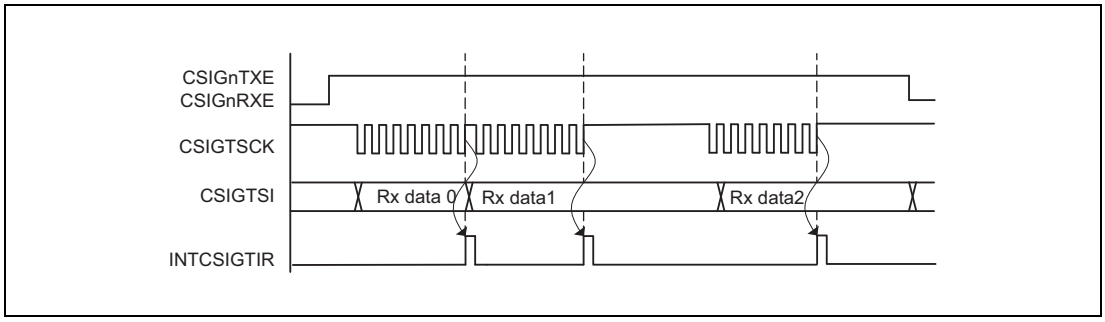



**Figure 16.5 Generation of INTCSIGTIR**

### 16.4.4 INTCSIGTIRE (Communication Error Interrupt)

This interrupt is generated whenever an error is detected.

**Table 16.25 Data Error Types**

| Error Type | Communication Status After Error Interrupt | Note |
|---|---|---|
| Parity error | Interrupt is generated and communication continues. | — |
| Data consistency check error | Interrupt is generated and communication continues. | — |
| Overrun error*1 | When CSIGnCTL1.CSIGnHSE = 0 (handshake function disabled) in slave mode, interrupt is generated and communication continues. | When CSIGnCTL1.CSIGnHSE = 1 (handshake function enabled) in slave mode, communication stops due to the handshake. An interrupt is not generated and an overrun error does not occur. |

Note 1. In master mode, overrun errors do not occur.
In slave mode, communication cannot be stopped.

The type of error that caused the generation of INTCSIGTIRE is indicated in register CSIGnSTR0.

For details about the various error types, see **Section 16.5.10, Error Detection**.

# 16.5 Operation

## 16.5.1 Master/Slave Mode

Whether CSIG operates in master mode or in slave mode depends on the setting of bits CSIGnCTL2.CSIGnPRS[2:0]. If master mode is selected, the source of the transmission clock must be selected as well.

### 16.5.1.1 Master Mode

In master mode, the serial transmission clock is generated by the built-in baud rate generator (BRG) and supplied to the slave via signal CSIGTSCK.

Master mode is enabled by setting bits CSIGnCTL2.CSIGnPRS[2:0] to values other than $111_B$. In master mode, the frequency of BRG can be configured by setting bits CSIGnCTL2. CSIGnPRS[2:0] and bits CSIGnCTL2.CSIGnBRS[11:0].

The default level of CSIGTSCK depends on the CSIGTSCK clock inversion function bit; it is high when CSIGnCTL1.CSIGnCKR = 0, and is low when CSIGnCTL1.CSIGnCKR = 1.

The example below shows the communication in master mode for 8-bit data, CSIGnCTL1.CSIGnCKR = 0, CSIGnCFG0.CSIGnDAP = 0, and MSB first:

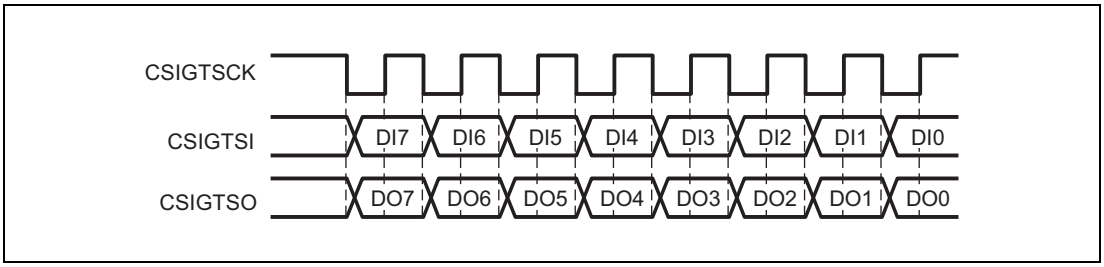



**Figure 16.6 Transmission/Reception in Master Mode**

### 16.5.1.2 Slave Mode

In slave mode, another device is the communication master. The external clock is supplied via the signal CSIGTSCK. Transmit/receive operation starts as soon as a clock signal is detected.

Slave mode is selected by setting CSIGnCTL2.CSIGnPRS[2:0] to $111_B$.

**NOTE**

When using slave mode, disable the baud rate generator (BRG) by setting bits CSIGnCTL2.CSIGnBRS[11:0] to $000_H$.

The example below shows the communication in slave mode for 8-bit data, CSIGnCTL1.CSIGnCKR = 0, CSIGnCFG0.CSIGnDAP = 0, and MSB first:

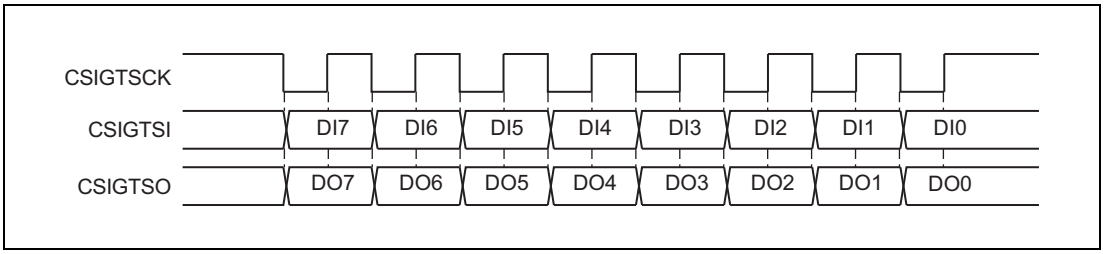



**Figure 16.7 Transmission/Reception in Slave Mode**

## 16.5.2 Master/Slave Connections

### 16.5.2.1 One Master and One Slave

The following figure illustrates the connections between one master and one slave.

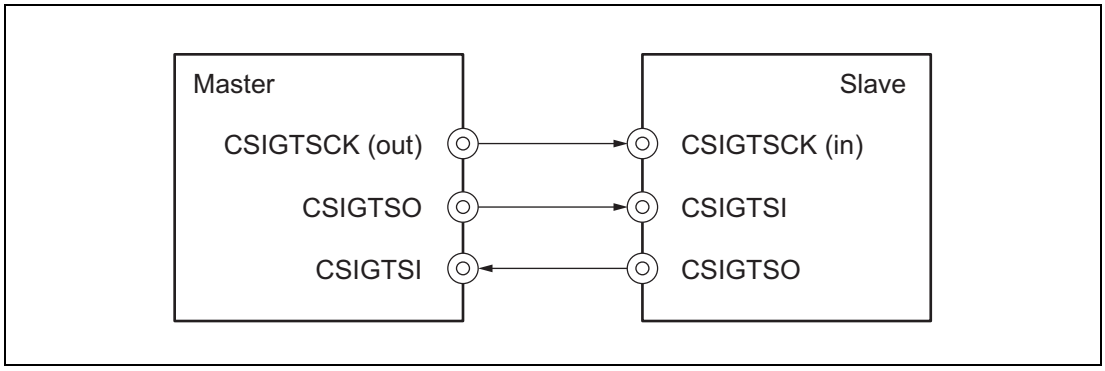



**Figure 16.8 Direct Master/Slave Connection**

### 16.5.2.2 One Master and Multiple Slaves

The following figure illustrates the connections between one master and multiple slaves. In this case, the master must provide one slave select (SS) signal to each of the slaves. This signal is connected to the slave select input $\overline{\text{CSIGTSSI}}$ of the slave.

The $\overline{\text{CSIGTSSI}}$ signal can be enabled or disabled by the CSIGnCTL1.CSIGnSSE bit.

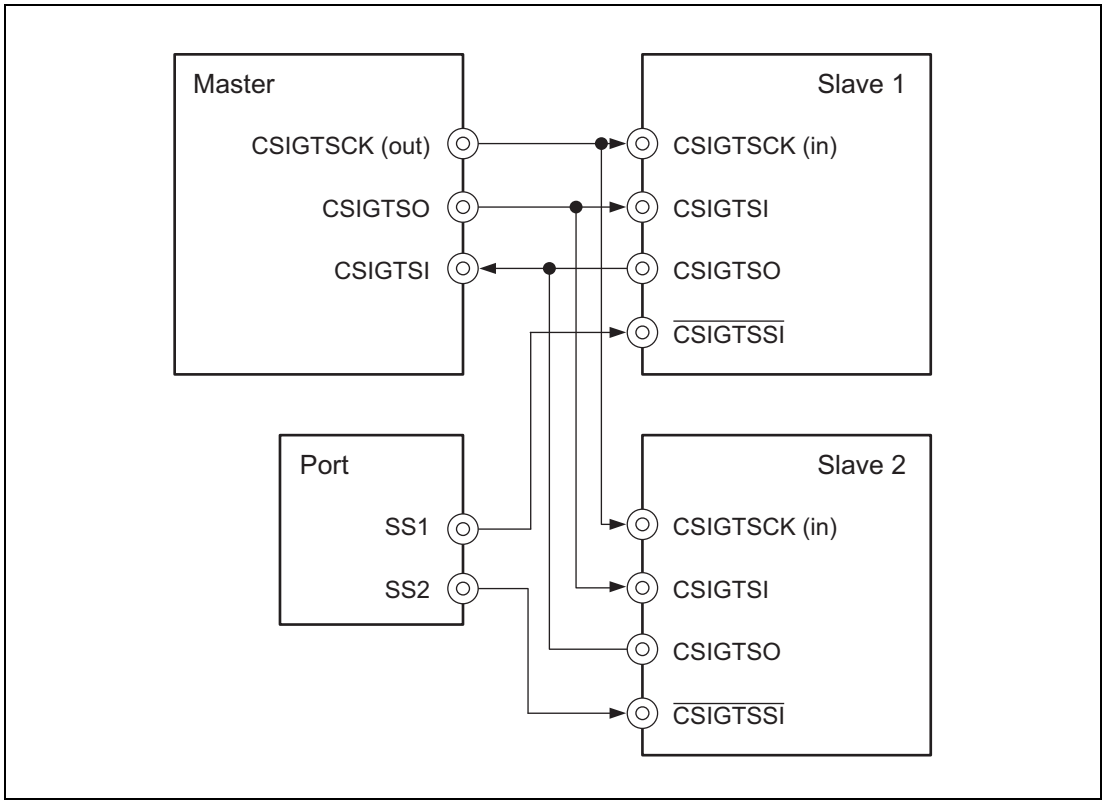



**Figure 16.9 Master to Multiple Slaves Connection**

A slave is selected (enabled) when its $\overline{\text{CSIGTSSI}}$ signal is low.

If it is not selected, the slave will neither receive nor transmit data. In addition, when transmit-only mode or transmit/receive mode is set (CSIGnCTL0.CSIGnTXE = 1), the CSIGTSO output buffer of the slave which is not selected is disabled and set to input mode in order to avoid interference with the outputs of other selected slaves.

### 16.5.3 Transmission Clock Selection

In master mode, the transfer clock frequency is selectable using the CSIGnPRS[2:0] and CSIGnBRS[11:0] bits in the CSIGnCTL2 register.

The following figure shows a block diagram of the BRG.

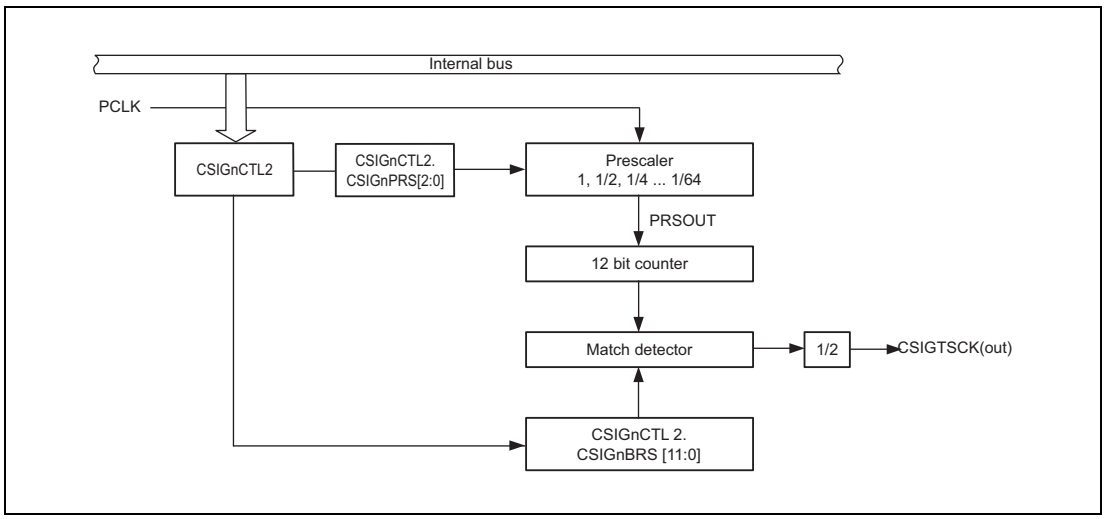



**Figure 16.10 BRG Block Diagram**

Setting CSIGnCTL2.CSIGnBRS[11:0] to $000_H$ disables the BRG.

#### Transfer clock frequency calculation

The transfer clock frequency in master mode is calculated as:

Transfer clock frequency (CSIGTSCK) = PCLK / (division ratio of PCLK) = PCLK / ($2^{\alpha} \times k \times 2$)
where:

$\alpha$ = CSIGnCTL2.CSIGnPRS[2:0] = 0 to 6

k = CSIGnCTL2.CSIGnBRS[11:0] = 1 to 4095

#### Transfer clock frequency upper and lower limits

When specifying the transfer clock frequency, please note the following:

- For the maximum transfer clock frequency of this product in master mode or slave mode, refer to the CSIG timing shown in the electrical characteristics. In addition, in either mode, specify a frequency within the defined range.
- The minimum transfer clock frequency in master mode and slave mode is PCLK/524160.
- The maximum transfer clock frequency is as follows:
  - In master mode: 10.0 MHz (however, it must be equal to or lower than PCLK/4)
  - In slave mode: 5.0 MHz (however, it must be equal to or lower than PCLK/16)

## 16.5.4 Data Transfer Modes

### 16.5.4.1 Transmit-Only Mode

Setting CSIGnCTL0.CSIGnTXE = 1 and CSIGnCTL0.CSIGnRXE = 0 places CSIG in transmit-only mode. Transmission starts when data to be transmitted is written in the CSIGnTX0W or CSIGnTX0H register.

**CAUTION**

**When the mode transitions from one of the receive modes to transmit-only mode, the data in the CSIGnRX0 buffer becomes undefined after completion of the first transmission.**
**Therefore, the reception register CSIGnRX0 has to be read before switching to transmit-only mode.**

### 16.5.4.2 Receive-Only Mode

Setting CSIGnCTL0.CSIGnTXE = 0 and CSIGnCTL0.CSIGnRXE = 1 puts the CSIG in receive-only mode.

In master mode, reception starts when dummy data is read from the CSIGnRX0 register.
All subsequent receptions are triggered by reads from the CSIGnRX0 register, as long as CSIGnBCTL0.CSIGnSCE = 1.

Moreover, CSIGnBCTL0.CSIGnSCE has to be set to 0 before reading the last received data from CSIGnRX0.

The recommended procedure is:

1. Set CSIGnBCTL0.CSIGnSCE = 1.
2. Read CSIGnRX0 (dummy data).
3. Wait for the reception interrupt INTCSIGTIR.
4. Read CSIGnRX0 (received data).
   In case more data receptions follow at step 3, continue to read until all data is received.
   Before reading the last received data from CSIGnRX0, set CSIGnBCTL0.CSIGnSCE = 0.

In slave mode, reception starts when the communication clock CSIGTSCK is received from the master. In this case, it is not necessary to read data from the CSIGnRX0 register of the slave.

**NOTE**

In slave mode, any previously received data must be read from the reception register CSIGnRX0 in order to avoid any overwrite situation.

### 16.5.4.3 Transmit/Receive Mode

Setting CSIGnCTL0.CSIGnTXE = 1 and CSIGnCTL0.CSIGnRXE = 1 puts the CSIG in transmit/receive mode.

Data transfer (transmission and reception) starts when the data to be transmitted is written to the CSIGnTX0W or CSIGnTX0H register.

## 16.5.5 Data Length Selection

### 16.5.5.1 Data Length Selection without Extended Length

Transmission data length is selectable from 7 to 16 bits using the CSIGnDLS[3:0] bits in the CSIGnCFG0 register. The examples below show the communication with MSB first (CSIGnCFG0.CSIGnDIR = 0):

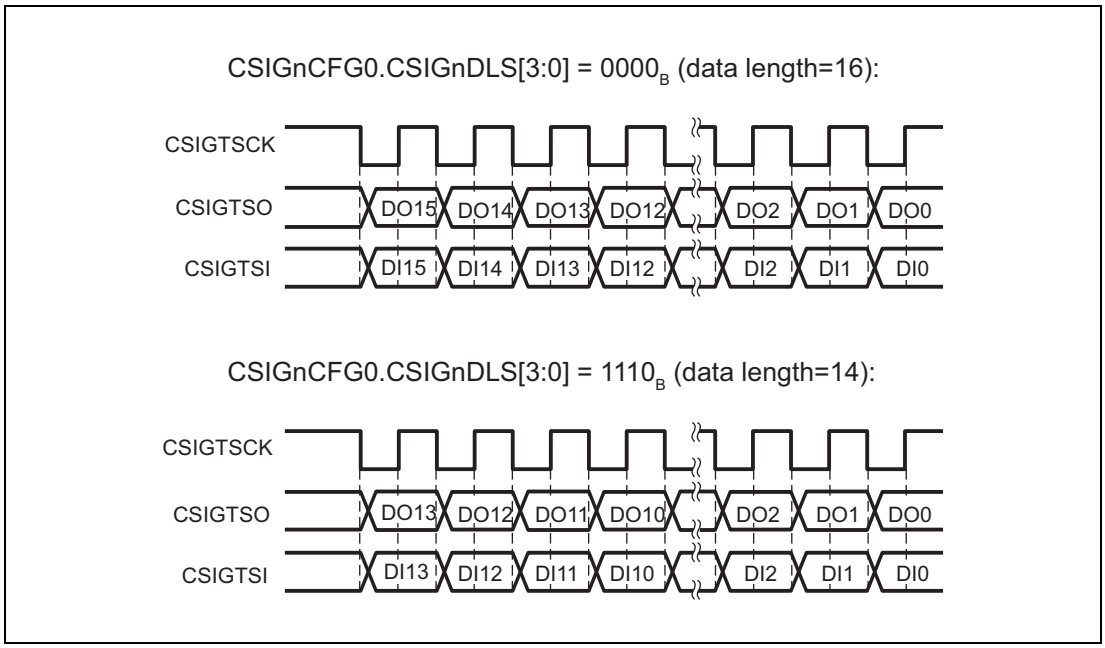



**Figure 16.11 Data Length Selection Function**

### 16.5.5.2 Data Length Selection with Extended Data Length

If the data to be transmitted/received exceeds 16 bits, the extended data length (EDL) function can be used.

The EDL function is enabled by setting the CSIGnCTL1.CSIGnEDLE bit to 1.

The following describes how the EDL function works and how to specify the EDL setting.

- The data has to be broken into 16-bit blocks plus remainder. For example, data of 42 bits would be broken into two 16-bit blocks plus 10 bits.
- The bit length of the remainder is set as “data length” in CSIGnCFG0.CSIGnDLS[3:0] bits.
- Set the CSIGnTX0W.CSIGnEDL bit to 1 to transmit the 16-bit blocks. In this case, the data written to the CSIGnTX0W register is sent as 16-bit data regardless of the setting of the CSIGnCFG0.CSIGnDLS[3:0] bits.
- The transfer completes after the data with the specified data length (the remainder when CSIGnTX0W.CSIGnEDL = 0) has been sent.

**Example**

Example of sending 40 bits of data, $123456789A_H$:

40 bits are split into 2 × 16 bits plus 8 bits.

- Initialize CSIGnCFG0.CSIGnDLS[3:0] = $8_D$.
- To transmit the data $123456789A_H$ with MSB first, write the following sequence to CSIGnTX0W:
  - 2000 $1234_H$ (CSIGnTX0W.CSIGnEDL = 1)
  - 2000 $5678_H$ (CSIGnTX0W.CSIGnEDL = 1)
  - 0000 $009A_H$ (CSIGnTX0W.CSIGnEDL = 0)

The following figure illustrates the timing.

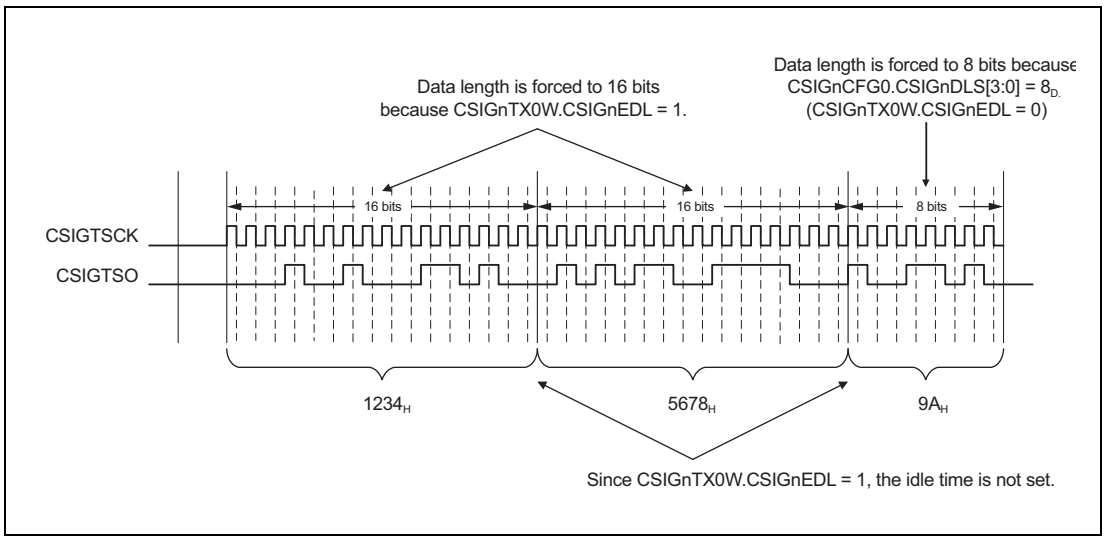



**Figure 16.12 EDL Timing Diagram**

**NOTES**

1. Data length with less than 7 bits can be set only when EDL mode is used.
2. It is not possible to transmit two consecutive data with a data length of less than 7 bits.
3. If parity is enabled, the parity bit is added after the last bit.
4. Example for setting the data direction:
   - Data to be transmitted: $123456_H$
   - MSB first:
     Set CSIGnCFG0.CSIGnDIR to 0.
     Write 2000 $1234_H$ to CSIGnTX0W (EDL bit = 1).
     Write 0000 $0056_H$ to CSIGnTX0W (EDL bit = 0).
   - LSB first:
     Set CSIGnCFG0.CSIGnDIR to 1.
     Write 2000 $3456_H$ to CSIGnTX0W (EDL bit = 1).
     Write 0000 $0012_H$ to CSIGnTX0W (EDL bit = 0).
5. EDL mode cannot be used in the slave mode configured for receive-only mode. (CSIGnCTL2.CSIGnPRS[2:0] = $111_B$, CSIGnCTL0.CSIGnTXE = 0, CSIGnCTL0.CSIGnRXE = 1)

### 16.5.6 Serial Data Direction Selection Function

The serial data direction is selectable using the CSIGnDIR bit in the CSIGnCFG0 register. The examples below show the communication for 8-bit data (CSIGnCFG0.CSIGnDLS[3:0] = $1000_B$):

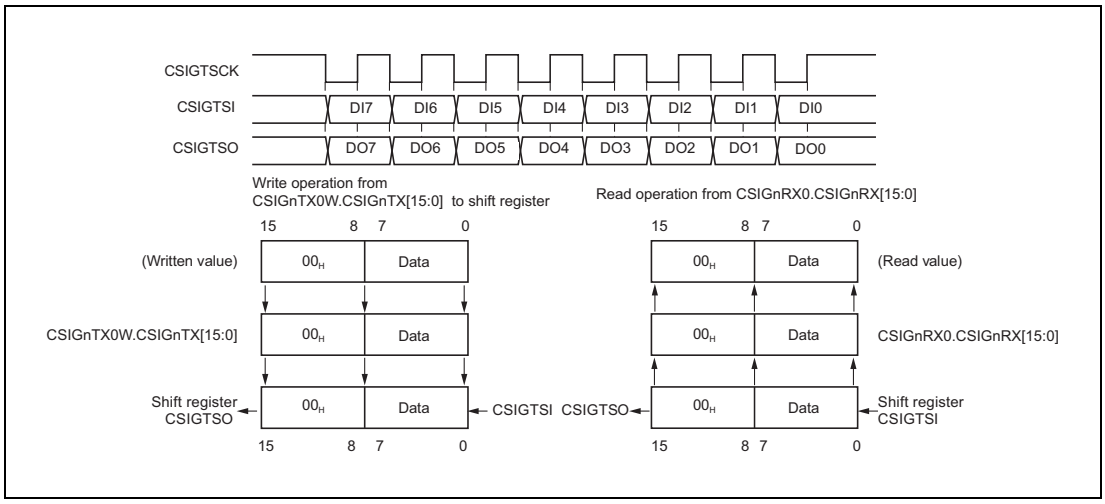



**Figure 16.13 Serial Data Direction Select Function — MSB First (CSIGnDIR = 0)**

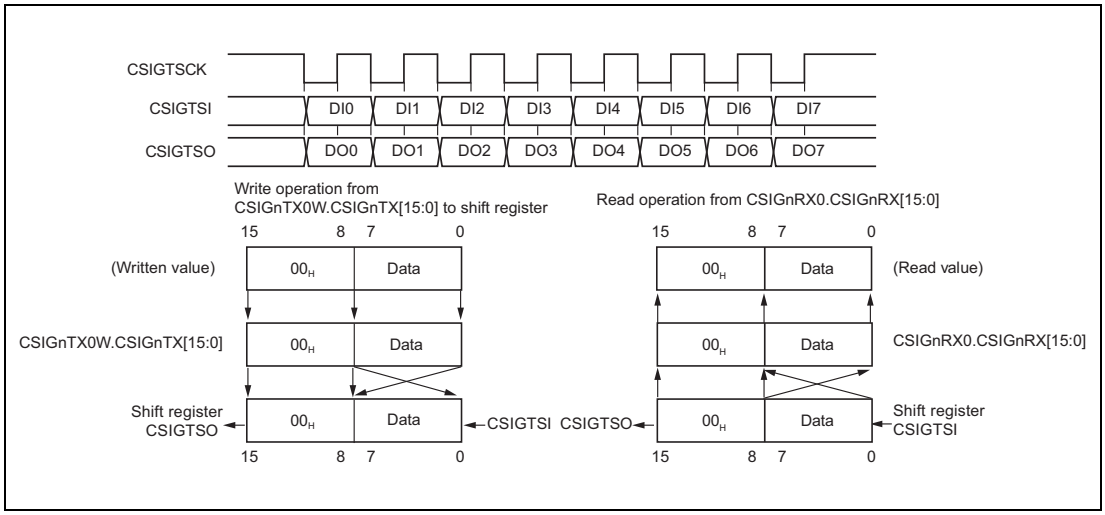



**Figure 16.14 Serial Data Direction Select Function — LSB First (CSIGnDIR = 1)**

### 16.5.7 Communication in Slave Mode

The following figure illustrates the communication signals and timing in slave mode.

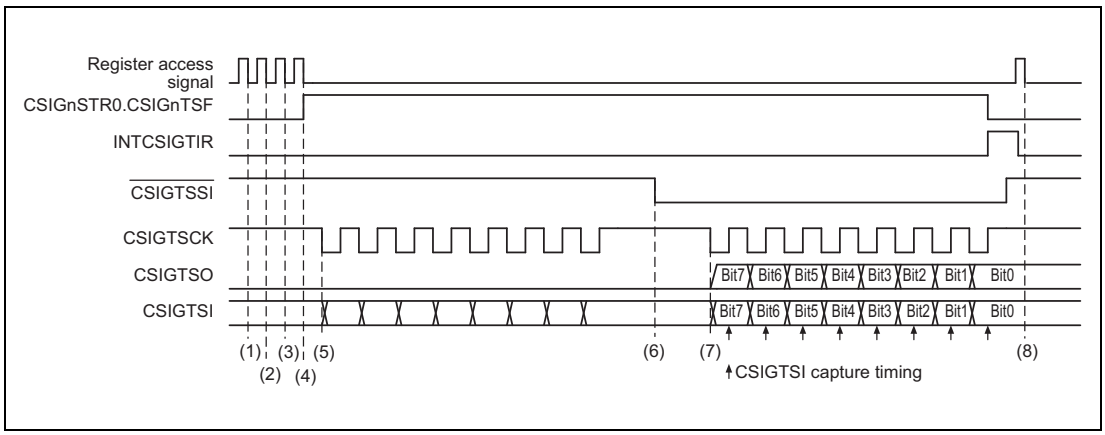



**Figure 16.15 Rx/Tx Communication Timing in Slave Mode**

1. Slave mode is selected (CSIGnCTL2.CSIGnPRS[2:0] = $111_B$), the $\overline{\text{CSIGTSSI}}$ signal is enabled (CSIGnCTL1.CSIGnSSE = 1) and the clock phase is set to the high level (CSIGnCTL1.CSIGnCKR = 1).
2. Data length is 8 bits (CSIGnCFG0.CSIGnDLS[3:0] = $1000_B$).
   Data direction is set to MSB first (CSIGnCFG0.CSIGnDIR = 0).
3. CSIG is set to transmit/receive mode (CSIGnCTL0.CSIGnPWR = 1, CSIGnCTL0.CSIGnTXE = 1, CSIGnCTL0.CSIGnRXE = 1).
4. When transfer data is written to the transmission register CSIGnTX0H, the transfer status flag CSIGnSTR0.CSIGnTSF is automatically set and the CSIG waits until signal $\overline{\text{CSIGTSSI}}$ goes low.
5. While signal $\overline{\text{CSIGTSSI}}$ is high, transmission/reception is not started even if the serial clock is input. CSIGTSO retains the values and input at CSIGTSI is ignored.
6. As soon as $\overline{\text{CSIGTSSI}}$ falls to low level, CSIGTSO is enabled.
7. If the serial clock is input to the CSIG while $\overline{\text{CSIGTSSI}}$ is low, transfer data is sent to CSIGTSO in synchronization with the serial clock, and simultaneously data is received from CSIGTSI.
8. The register CSIGnRX0 is read.

### 16.5.8 Handshake Function

CSIG features a handshake function to synchronize the master and the slave devices. This function can be enabled/disabled by the CSIGnCTL1.CSIGnHSE bit. For handshake, the signals CSIGTRYI and CSIGTRYO are used.

The busy timing depends on the setting of the data phase selection bit CSIGnCFG0.CSIGnDAP.

#### 16.5.8.1 Slave Mode

If CSIGnCTL1.CSIGnHSE = 1, a low-level CSIGTRYO signal is output when the slave becomes busy. This happens when previously received data is still in the CSIGnRX0 register, and the new data cannot be copied from the shift register to CSIGnRX0 (CSIGnRX0 full condition).

The following examples assume a data length of 8 bits.

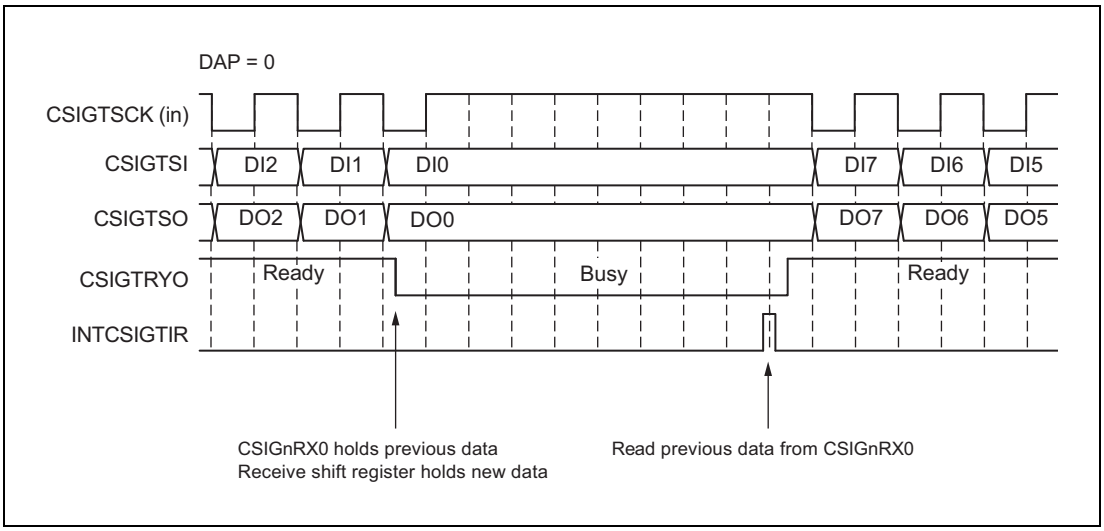



**Figure 16.16 Ready/Busy Signal from Slave (CSIGnCFG0.CSIGnDAP = 0)**

While the slave is busy, the master has to wait (i.e. suspend the transmission clock). The slave sets CSIGTRYO to high (“ready”) as soon as the read from the reception register CSIGnRX0 completes.

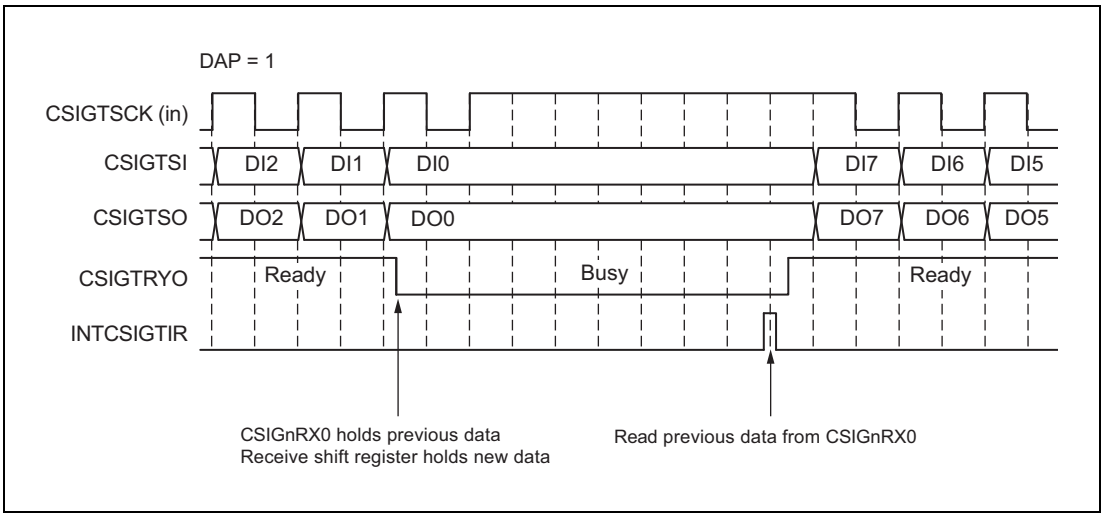



**Figure 16.17 Ready/Busy Signal from Slave (CSIGnCFG0.CSIGnDAP = 1)**

### 16.5.8.2 Master Mode

When the master detects low level of the CSIGTRYI while CSIGnCTL1.CSIGnHSE = 1, subsequent transfers are put on hold, the master goes into wait state and suspends the CSIGTSCK clock.

The CSIGTRYI level is checked at each half clock cycle of CSIGTSCK.

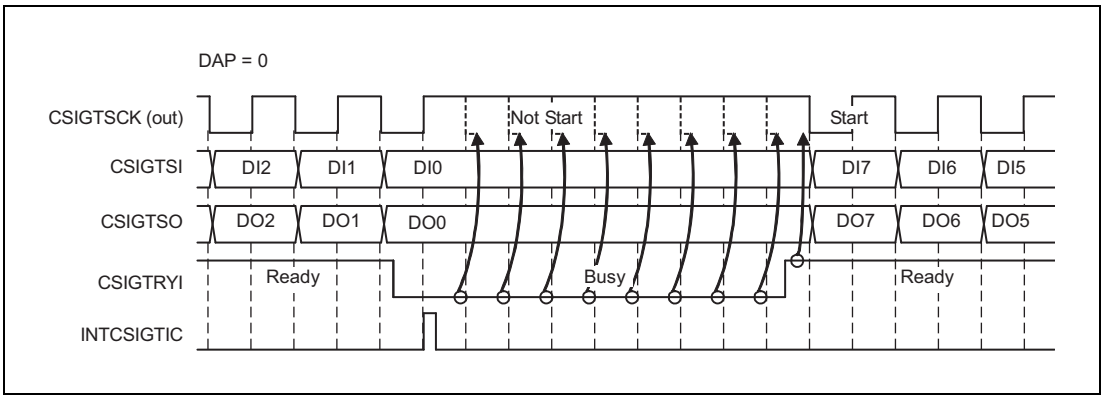



**Figure 16.18 Master's Reaction to CSIGTRYI (CSIGnCFG0.CSIGnDAP = 0)**

If the CSIGTRYI low signal from the slave is received while data transfer is in progress, the serial clock is suspended after the transfer is complete.

The master resumes the communication as soon as CSIGTRYI becomes high (slave is "ready").

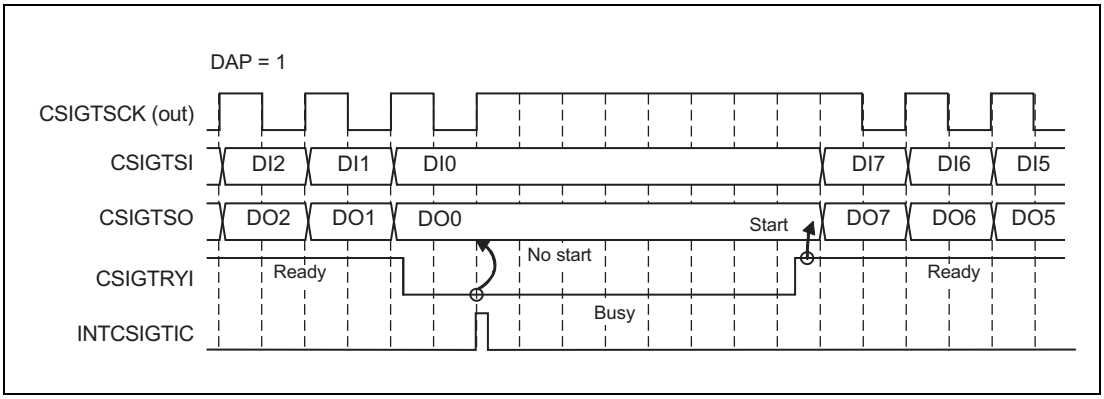



**Figure 16.19 Master's Reaction to CSIGTRYI (CSIGnCFG0.CSIGnDAP = 1)**

**CAUTION**

**If multiple slaves are connected, the master must only detect the CSIGTRYI signal of the slave it has selected for communication.**

**CSIGTRYI of the master must be pulled down by the slave before the next transfer starts. Even if the signal is pulled down by the slave during the transfer, the transfer will continue until it completes.**

### 16.5.9 Loop-Back Mode

Loop-back mode is a special mode for self-test. This feature is only available in master mode.

When this mode is active (CSIGnCTL1.CSIGnLBM = 1), the transmit and receive signals are internally connected, as shown in the figures below. The signals CSIGTSCK, CSIGTSO, and CSIGTSI are disconnected from the ports. In addition, the CSIGTSO output level is fixed to low, and CSIGTSCK is set to reset level (High). The rest of CSIG works as in normal operation.

In order to test the CSIG, set the loop-back mode and carry out normal transfer operations. Then check that the received data is the same as the transmitted data.

**Table 16.26 Output Level of Pins**

| Pin | Output Level |
|---|---|
| CSIGTSCK(out) | High level |
| CSIGTSO | Low level (not dependent on the previous value) |
| Interrupt | Normal function |
| CSIGTRYO | Normal function (Low level) |

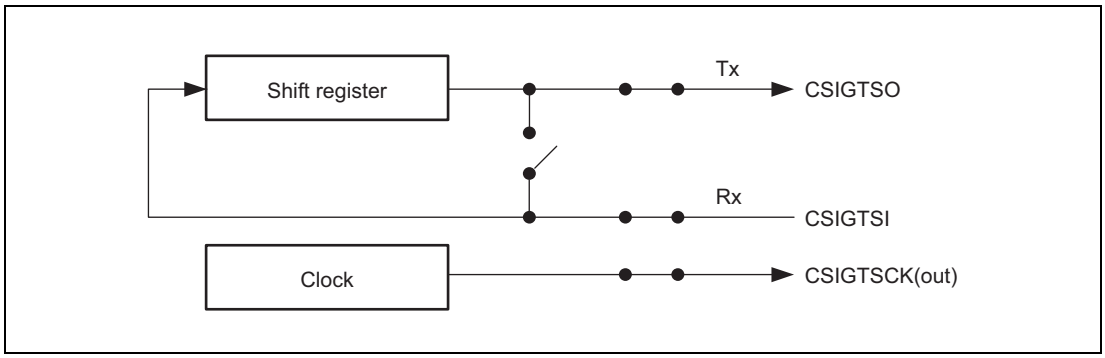



**Figure 16.20 Normal Operation**

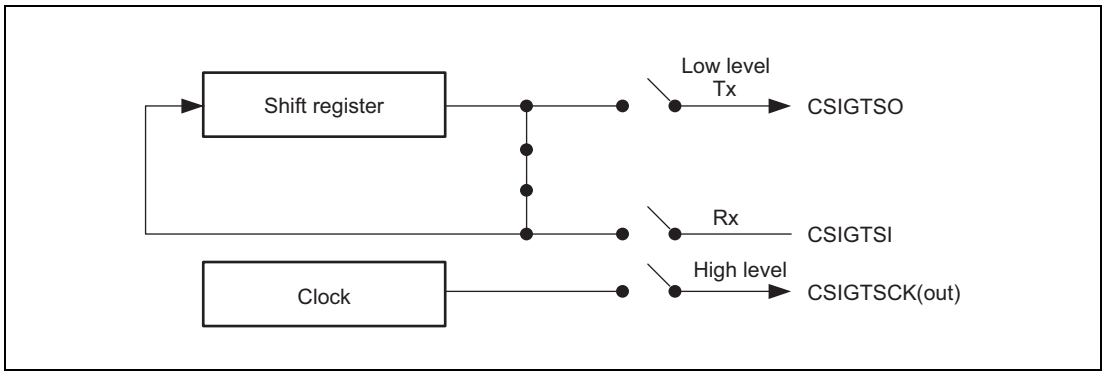



**Figure 16.21 Operation in Loop-Back Mode**

### 16.5.10 Error Detection

CSIG can detect three error types:

- Data consistency check error (transmission data)
- Parity error (received data)
- Overrun error (received data)

Data consistency check error and parity error check functions can be individually enabled or disabled.

If one of these errors is detected, the interrupt INTCSIGTIRE is generated.

#### 16.5.10.1 Data Consistency Check

The purpose of the data consistency check is to ensure that the data physically sent as an output signal is identical to the original data that was copied to the shift register.

The data consistency check can be enabled/disabled by the CSIGnCTL1.CSIGnDCS bit (when checking data consistency, make sure that PIPCn.PIPCn_m = 1 for CSIGTSO). It will not be enabled if data transmission is disabled (CSIGnCTL0.CSIGnTXE = 0).

When the data consistency check is enabled, the data transferred from CSIGnTX0W or CSIGnTX0H to the shift register is copied to a separate register. In addition, the physical levels of CSIGTSO are captured and their logical interpretation is written to the corresponding shift register.

After completion of the transmission, the data sent is compared with the original transmission data.

Mismatch is considered as a data consistency check error and:

- Interrupt INTCSIGTIRE is generated.
- The CSIGnSTR0.CSIGnDCE bit is set.

The data consistency check function is illustrated in the following block diagram.

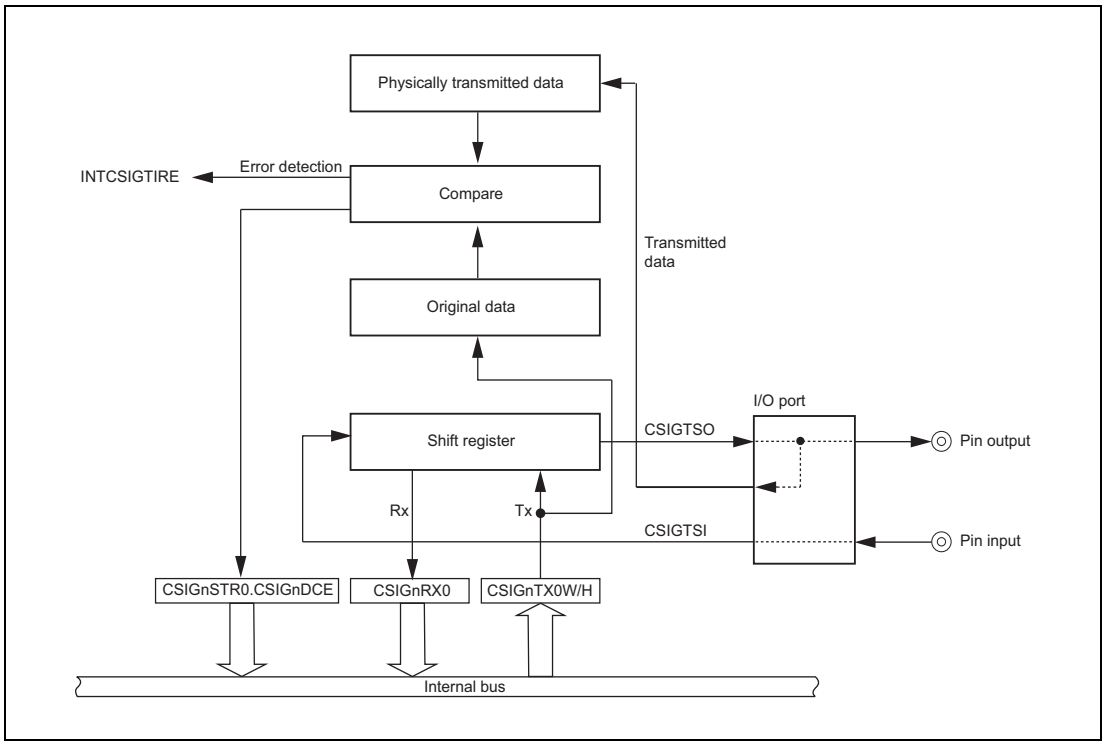



**Figure 16.22 Functional Block Diagram of the Data Consistency Check**

### 16.5.10.2 Parity Check

Parity is a common mean to detect a single bit error during data transmission. CSIG can append a parity bit after the last data bit (even if extended data length is used).

The use and type of parity is specified by CSIGnCFG0.CSIGnPS[1:0].

Parity check is enabled if CSIGnCFG0.CSIGnPS[1] = 1.

The parity bit is checked after reception is complete. If a parity error occurs:

- Interrupt INTCSIGTIRE is generated.
- The CSIGnSTR0.CSIGnPE bit is set.

The following figure shows an example.

Data length is 8 bits. The data transmitted is $05_H$ and $35_H$. Parity type is odd.

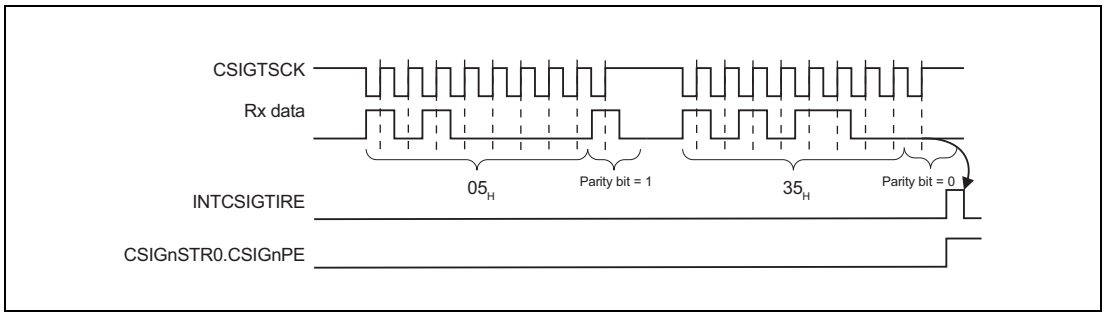



**Figure 16.23 Parity Check Example**

For the first 8 bits, the parity bit is 1. There is no parity error, because the total number of ones (including the parity bit) is odd.

For the second 8 bits, the parity bit is 0. This is detected as a parity error, because the total number of ones (including the parity bit) is even.

If the EDL (extended data length) function is used, the parity bit is added after the last data bit.

### 16.5.10.3 Overrun Error

This error occurs when previously received data still resides in the reception register CSIGnRX0 because it has not been read, and new data is received.

The overrun error is not generated if data reception is disabled (CSIGnCTL0.CSIGnRXE = 0).

If overrun occurs:

- Interrupt INTCSIGTIRE is generated
- The CSIGnSTR0.CSIGnOVE bit is set
- Data in the CSIGnRX0 register is overwritten and communication continues.

The following figure illustrates the overrun error detection function.

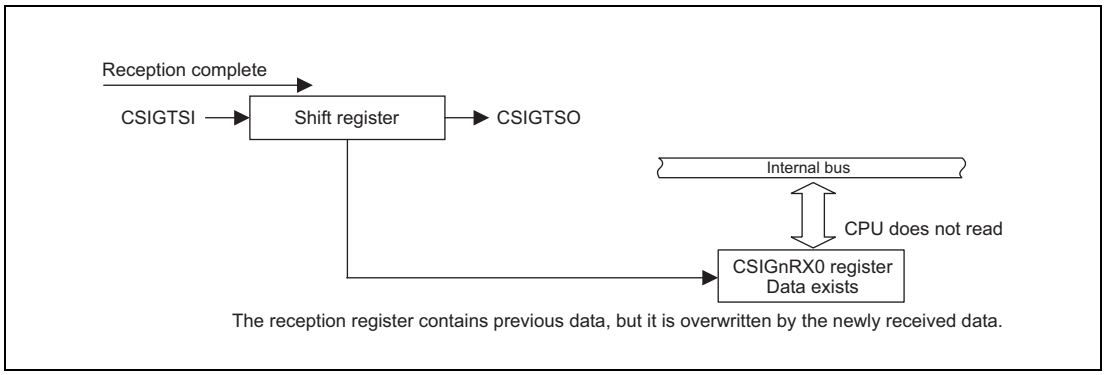



**Figure 16.24 Overrun Error Detection**

The following figure illustrates an example where:

- Rx data 3 is not read
- Rx data 4 is received, and data is overwritten.

Thus an overrun error occurs.

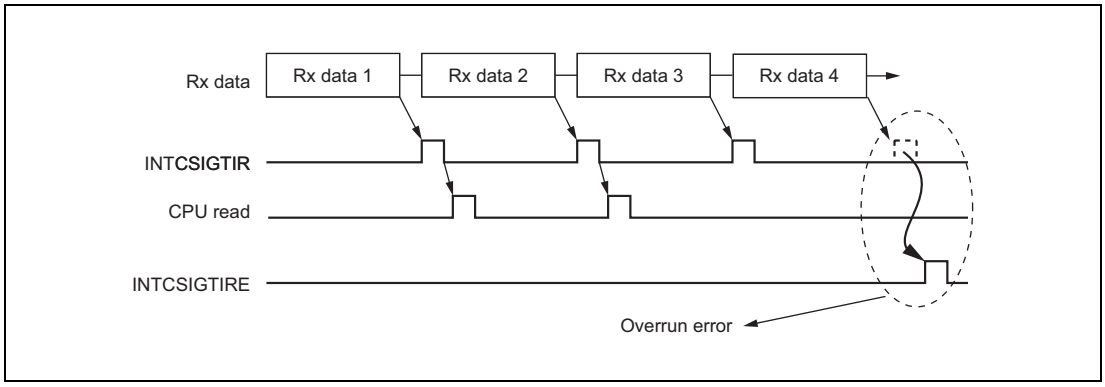



**Figure 16.25 Overrun Error Detection - Example**

**NOTE**

An overrun error can be avoided by using the handshake.

When handshake is used in slave mode, the receiver (slave) signals to the transmitter (master) that it is busy. The transmitter then waits until the receiver reads its reception register and becomes ready again.

For details see **Section 16.5.8, Handshake Function**.

# 16.6 Operating Procedures

## 16.6.1 Master Mode Transmission/Reception by DMA

This section describes an example of performing the transmission/reception in master mode in combination with a DMA.

The following instructions are based on the assumption that:

- Transmission data length is 8 bits (CSIGnCFG0.CSIGnDLS[3:0] = $1000_B$)
- MSB is transmitted first (CSIGnCFG0.CSIGnDIR = 0)
- INTCSIGTIC interrupt is generated at the end of the transfer (CSIGnCTL1.CSIGnSLIT = 0)
- Normal clock and data phase (CSIGnCTL1.CSIGnCKR = 0, CSIGnCFG0.CSIGnDAP = 0)
- The number of data is 10 (0 to 9)

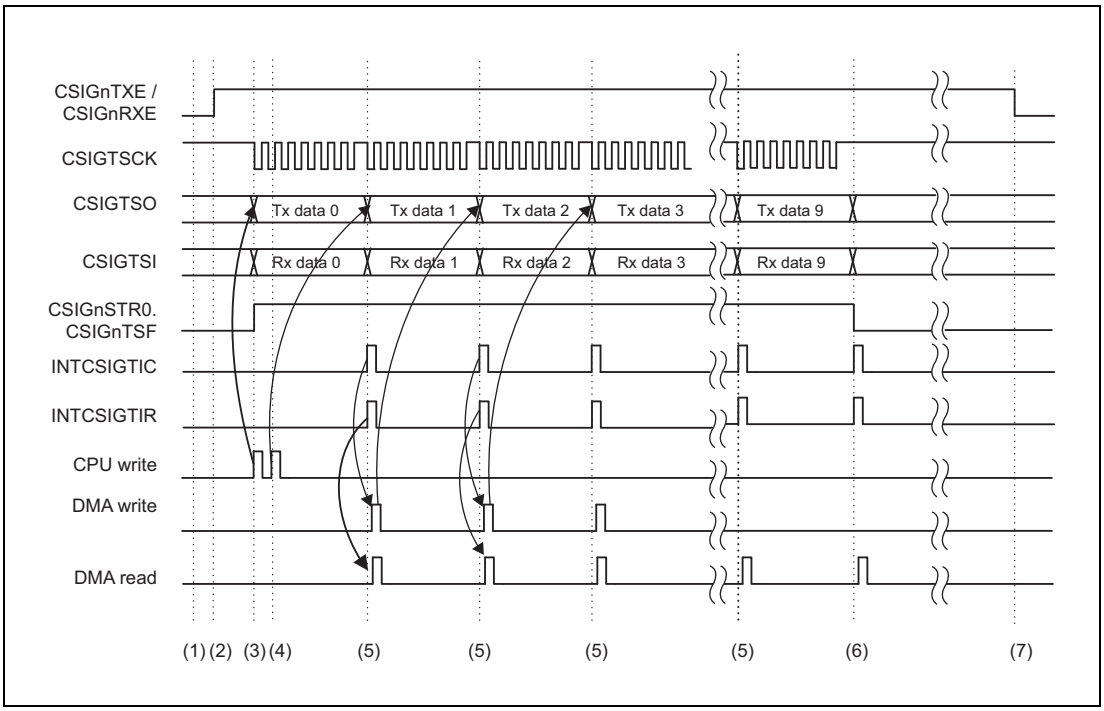



**Figure 16.26 Communication in Master Mode**

**Procedure:**

1. Configure the communication protocol in register CSIGnCFG0. Specify the interrupt timing and operation mode by setting the corresponding bits of the CSIGnCTL1 register and CSIGnCTL2 register.
2. In the CSIGnCTL0 register, set CSIGnPWR = 1 (enable clock), CSIGnTXE = 1 (enable transmission), and CSIGnRXE = 1 (enable reception).
3. Write the first data to be sent to the transmission register CSIGnTX0H. Transmission starts automatically when the first data becomes available.
4. Write the second data to CSIGnTX0H. Writing the second data immediately after the first one avoids unnecessary delays between the data.

5. After the transmission or reception of each data, INTCSIGTIC or INTCSIGTIR interrupt is generated. INTCSIGTIC indicates that the next data can be written to CSIGnTX0H. INTCSIGTIR indicates that the reception register CSIGnRX0 must be read.
   In this example, CPU write and DMA write are assumed as equivalent.
6. No more write action is required after the transmission of data 8 completes. Data 9 (the last data) has been written after the transmission of data 7. However, the reception register CSIGnRX0 must be read after the reception of data 8 and 9 completes.
7. Finally, to disable the transmit/receive operation, clear CSIGnCTL0.CSIGnTXE and CSIGnCTL0.CSIGnRXE. When no communication is taking place, set CSIGnCTL0.CSIGnPWR to “0” to minimize the power consumption of the CSIGn.

# Section 17 Clocked Serial Interface H (CSIH)

This section contains a generic description of the Clocked Serial Interface H (CSIH).

The first part in this section describes the features specific to RH850/F1K, such as the number of units, register base addresses, etc. The remainder of the section describes the functions and registers of CSIH.

## 17.1 Features of RH850/F1K CSIH

### 17.1.1 Number of Units

This microcontroller has the following number of CSIH units.

Each CSIH unit has one channel interface.

**Table 17.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of units | 4 | 4 | 4 |
| Name | CSIHn (n = 0 to 3) | CSIHn (n = 0 to 3) | CSIHn (n = 0 to 3) |

**Table 17.2 Indices**

| Index | Description |
|---|---|
| n | Throughout this section, the individual CSIH units are identified by the index “n” (n = 0 to 3): for example, CSIHnCTL0 is the CSIHn control register 0. |
| x | CSIHn has a maximum of 8 chip select signals. Throughout this section, the individual chip select signals are identified by the index “x”: that is, CSx denotes a non-specified chip select signal. |
| y | A variable used for explanation is identified by the index “y”: for example, CSIHnBRSy is the baud rate setting register of CSIHn. |

The following table shows values indicated by the indices of each product.

**Table 17.3 Indices of Products**

| Indices of Each Product | | |
|---|---|---|
| **100 pins** | **144 pins** | **176 pins** |
| For the value of x, see **Table 17.4, Number of Chip Select Signals**. | | |
| y = 0 to 3 | | |

The numbers of chip select signals for each of the CSIH units are listed in the following table.

**Table 17.4 Number of Chip Select Signals**

| Unit Name | Chip Select Index | | |
|---|---|---|---|
| | 100 pins | 144 pins | 176 pins |
| CSIH0 | CSx (x = 0 to 7) | CSx (x = 0 to 7) | CSx (x = 0 to 7) |
| CSIH1 | CSx (x = 0 to 5) | CSx (x = 0 to 5) | CSx (x = 0 to 5) |
| CSIH2 | CSx (x = 0 to 3) | CSx (x = 0 to 5) | CSx (x = 0 to 5) |
| CSIH3 | CSx (x = 0 to 3) | CSx (x = 0 to 3) | CSx (x = 0 to 3) |

### 17.1.2 Register Base Address

CSIH base addresses are listed in the following table.

CSIH register addresses are given as offsets from the base addresses.

**Table 17.5 Register Base Addresses**

| Base Address Name | Base Address |
|---|---|
| <CSIH0_base> | FFD8 $0000_H$ |
| <CSIH1_base> | FFD8 $2000_H$ |
| <CSIH2_base> | FFD8 $4000_H$ |
| <CSIH3_base> | FFD8 $6000_H$ |

### 17.1.3 Clock Supply

The CSIH clock supply is shown in the following table.

**Table 17.6 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| CSIHn | PCLK | CKSCLK_ICSI | Communication clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CKSCLK_ICSI | |

### 17.1.4 Interrupt Requests

CSIH interrupt requests are listed in the following table.

**Table 17.7 Interrupt Requests**

| Unit Interrupt Name | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **CSIH0** | | | |
| INTCSIHTIC | Communication status interrupt | 29 | 70 |
| INTCSIHTIR | Receive status interrupt | 30 | 71 |
| INTCSIHTIRE | Communication error interrupt | 31 | — |
| INTCSIHTIJC | Job completion interrupt | 20, 32 | 72 |
| **CSIH1** | | | |
| INTCSIHTIC | Communication status interrupt | 16, 116 | 28 |
| INTCSIHTIR | Receive status interrupt | 17, 117 | 29 |
| INTCSIHTIRE | Communication error interrupt | 27, 118 | — |
| INTCSIHTIJC | Job completion interrupt | 28, 119 | 30 |
| **CSIH2** | | | |
| INTCSIHTIC | Communication status interrupt | 8, 132 | 89 |
| INTCSIHTIR | Receive status interrupt | 10, 133 | 90 |
| INTCSIHTIRE | Communication error interrupt | 11, 134 | — |
| INTCSIHTIJC | Job completion interrupt | 12, 135 | 91 |
| **CSIH3** | | | |
| INTCSIHTIC | Communication status interrupt | 9, 158 | 41 |
| INTCSIHTIR | Receive status interrupt | 13, 159 | 42 |
| INTCSIHTIRE | Communication error interrupt | 14, 160 | — |
| INTCSIHTIJC | Job completion interrupt | 15, 161 | 43 |

### 17.1.5 Reset Sources

CSIH reset sources are listed in the following table. CSIH is initialized by these reset sources.

**Table 17.8 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| CSIHn | All reset sources (ISORES) |

### 17.1.6 External Input/Output Signals

External input/output signals of CSIH are listed below.

**Table 17.9 External Input/Output Signals**

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **CSIH0** | | |
| CSIHTSCK | Serial clock signal | CSIH0SC*2 |
| CSIHTSI | Serial data input signal | CSIH0SI |
| $\overline{\text{CSIHTSSI}}$ | Slave select input signal | $\overline{\text{CSIH0SSI}}$ |
| CSIHTRYI | Ready/busy input signal | CSIH0RYI |
| CSIHTSO | Serial data output signal | CSIH0SO*2 |
| CSIHTRYO | Ready/busy output signal | CSIH0RYO |
| CSIHTCSS[7:0]*1 | Chip select signal | CSIH0CSS[7:0]*1 |
| **CSIH1** | | |
| CSIHTSCK | Serial clock signal | CSIH1SC*2 |
| CSIHTSI | Serial data input signal | CSIH1SI |
| $\overline{\text{CSIHTSSI}}$ | Slave select input signal | $\overline{\text{CSIH1SSI}}$ |
| CSIHTRYI | Ready/busy input signal | CSIH1RYI |
| CSIHTSO | Serial data output signal | CSIH1SO*2 |
| CSIHTRYO | Ready/busy output signal | CSIH1RYO |
| CSIHTCSS[5:0]*1 | Chip select signal | CSIH1CSS[5:0]*1 |
| **CSIH2** | | |
| CSIHTSCK | Serial clock signal | CSIH2SC*2 |
| CSIHTSI | Serial data input signal | CSIH2SI |
| $\overline{\text{CSIHTSSI}}$ | Slave select input signal | $\overline{\text{CSIH2SSI}}$ |
| CSIHTRYI | Ready/busy input signal | CSIH2RYI |
| CSIHTSO | Serial data output signal | CSIH2SO*2 |
| CSIHTRYO | Ready/busy output signal | CSIH2RYO |
| CSIHTCSS[5:0]*1 | Chip select signal | CSIH2CSS[5:0]*1 |
| **CSIH3** | | |
| CSIHTSCK | Serial clock signal | CSIH3SC*2 |
| CSIHTSI | Serial data input signal | CSIH3SI |
| $\overline{\text{CSIHTSSI}}$ | Slave select input signal | $\overline{\text{CSIH3SSI}}$ |
| CSIHTRYI | Ready/busy input signal | CSIH3RYI |
| CSIHTSO | Serial data output signal | CSIH3SO*2 |
| CSIHTRYO | Ready/busy output signal | CSIH3RYO |
| CSIHTCSS[3:0]*1 | Chip select signal | CSIH3CSS[3:0]*1 |

Note 1. For the number of chip select signals, see **Table 17.4, Number of Chip Select Signals**.

Note 2. For the port pins that are used as CSIHnSO and CSIHnSC, set the output driving ability to high (PDSCn_m = 1).

**CAUTION**

When port P8_6 is used as CSIH0CSS4, port P8_6 outputs a low-level $\overline{\text{RESETOUT}}$ signal while a reset is asserted and continues to output a low level after the reset is deasserted.

For details, see **Section 2.11.1.1, P8_6: $\overline{\text{RESETOUT}}$**.

### 17.1.7 Data Consistency Check

The port and the alternative function for data consistency check of CSIHnSO (CSIHTSO) output are shown in the following table. See **Section 17.5.12, Error Detection** for details on data consistency checking.

**Table 17.10 Port Pins for Data Consistency Checking**

| Unit Signal Name | Port Pin Name | Alternative Function |
|---|---|---|
| **CSIH0** | | |
| CSIHTSO | P0_3 | ALT_OUT4 |
| **CSIH1** | | |
| CSIHTSO | P0_5 | ALT_OUT3 |
| | P10_2 | ALT_OUT5 |
| **CSIH2** | | |
| CSIHTSO | P11_2 | ALT_OUT1 |
| **CSIH3** | | |
| CSIHTSO | P11_6 | ALT_OUT3 |

## 17.2 Overview

### 17.2.1 Functional Overview

- Three-wire serial synchronous data transfer
- Master mode or slave mode is selectable
- Multiple slaves configuration and RCB (Recessive Configuration for Broadcasting) are possible since there are to eight configurable chip select output signals
- Slave select input signal ($\overline{\text{CSIHTSSI}}$) is usable
- Built-in baud rate generator
- Transfer clock frequency is adjustable in master mode, whereas it is determined by the input clock in slave mode.
- Maximum transfer clock frequency:
  - Master mode: 10.0 MHz (however, it must be equal to or lower than PCLK/4)
  - Slave mode: 5.0 MHz (however, it must be equal to or lower than PCLK/16)
- Clock phases and data phases are selectable
- Data transfer with MSB first or LSB first is selectable
- Transfer data length is selectable from 2 to 16 bits in 1-bit units
- Built-in EDL (extended data length) function for transferring more than 16 bits of data
- Three selectable transfer modes:
  - transmit-only mode
  - receive-only mode
  - transmit/receive mode
- Built-in handshake function
- Built-in error detection (data consistency check, parity, time-out, overflow, and overrun)
- Support of job concept
- 128 words I/O buffer memory
- Direct access mode or memory mode (FIFO, dual buffer, and transmit-only buffer) is selectable
- Four different interrupt request signals (INTCSIHTIC, INTCSIHTIR, INTCSIHTIRE, INTCSIHTIJC)
- Built-in LBM (Loop Back Mode) function for self-test
- CPU-controlled high-priority communication function
- Enforced chip select idle setting
- Built-in RCB (Recessive Configuration for Broadcasting) bit
- Built-in JOB enable control bit for AUTOSAR

### 17.2.2 Functional Overview Description

The CSIH uses three signals for communication:

- Transmission clock CSIHTSCK (output in master mode, input in slave mode)
- Data output signal CSIHTSO
- Data input signal CSIHTSI

Additional signals are available for external control and monitoring:

- $\overline{\text{CSIHTSSI}}$: Slave select input signal
- CSIHTRYO: Ready/busy output signal (handshake signal)
- CSIHTRYI: Ready/busy input signal (handshake signal)
- CSIHTCSS[7:0]: Chip select signals

Data transmission is bit-wise and serial, and performed synchronously with the transmission clock.

The following table shows the most important registers for setting up the CSIH.

**Table 17.11 Main Registers of CSIH**

| Register | Function |
|---|---|
| CSIHnCTL0 | Enables/disables serial clock, and permits/ prohibits data transmission and data reception. Defines end-of-job behavior and enables/disables (bypasses) buffering. |
| CSIHnCTL1 | Controls options such as interrupt timing, extended data length, job feature, data consistency check, loop-back mode, handshake, etc. |
| CSIHnCTL2 | Selects master or slave mode, and the transfer clock frequency of the built-in baud rate generator (BRG) in master mode. |
| CSIHnBRSy | Specifies the transfer clock frequency for each chip select signal. |
| CSIHnMCTL0 | Selects memory mode and specifies the time-out value. |
| CSIHnMCTL1 | Controls the memory in FIFO mode. |
| CSIHnMCTL2 | Controls the memory in dual buffer mode. |
| CSIHnCFGx | Configures the communication protocol for each chip select signal. |

### 17.2.3 Block Diagram

The following block diagram shows the main components of the CSIH.

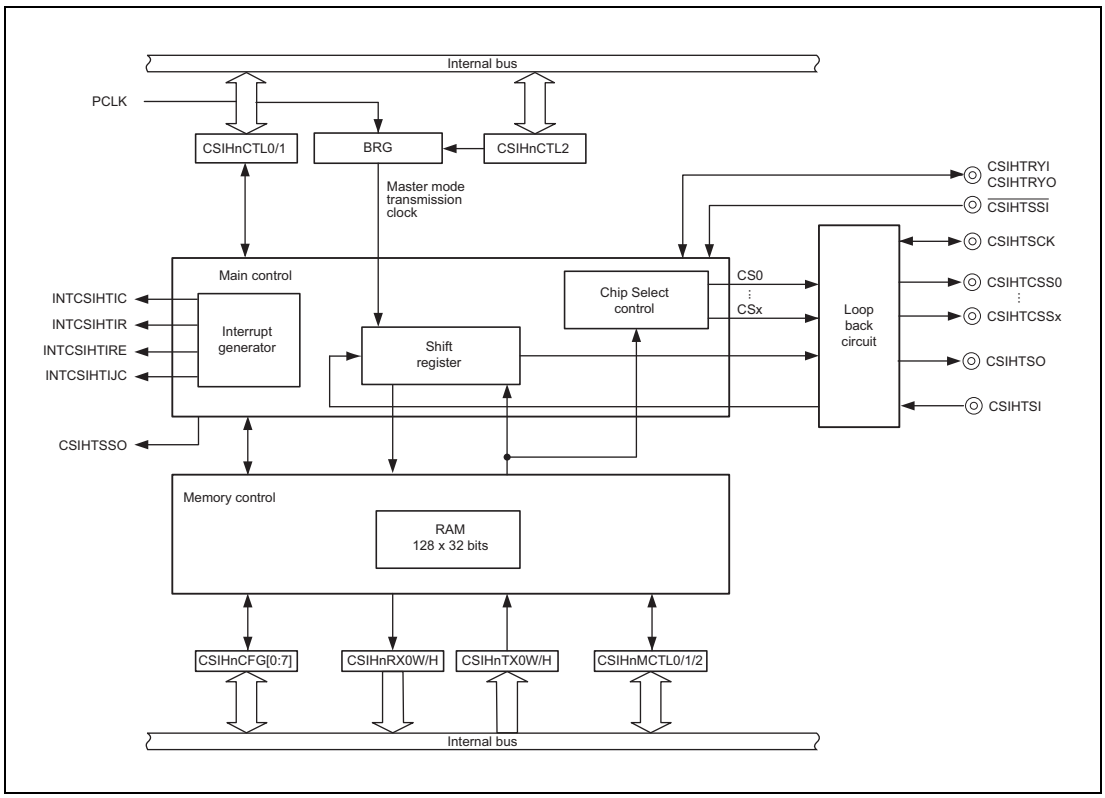



**Figure 17.1 CSIH Block Diagram**

In master mode, the transmission clock CSIHTSCK is generated by the built-in baud rate generator (BRG). In slave mode, the transmission clock is supplied by an external source.

The built-in memory can be configured as FIFO, dual buffer (separate transmit and receive buffers), or transmit-only buffer. It can also be bypassed for data transmission and reception without buffering.

The loop back circuit disconnects the CSIH completely from the ports and supports internal self-test.

**NOTE**

This section describes the following modes:

- The “operating mode” is either master mode or slave mode. The master can control and communicate with several slaves only in master mode (for details, see **Section 17.5.1, Operating Modes (Master/Slave)**).
- The “job mode” is related to the AUTOSAR job concept (for details, see **Section 17.5.3.3, Job Concept**).
- The “memory mode” accords with the various configurations of the associated buffer memory (for details, see **Section 17.5.6, CSIH Buffer Memory**).
- The “data transfer mode” specifies the type of communication – transmit-only, receive-only, or transmit/receive (for details, see **Section 17.5.7, Data Transfer Modes**).

# 17.3 Registers

## 17.3.1 List of Registers

CSIH registers are listed in the following table.

For details about <CSIHn_base>, see **Section 17.1.2, Register Base Address**.

**Table 17.12 List of Registers**

| Module | Register | Symbol | Address |
|---|---|---|---|
| CSIHn | CSIHn control register 0 | CSIHnCTL0 | <CSIHn_base> + $0000_H$ |
| | CSIHn control register 1 | CSIHnCTL1 | <CSIHn_base> + $0010_H$ |
| | CSIHn control register 2 | CSIHnCTL2 | <CSIHn_base> + $0014_H$ |
| | CSIHn status register 0 | CSIHnSTR0 | <CSIHn_base> + $0004_H$ |
| | CSIHn status clear register 0 | CSIHnSTCR0 | <CSIHn_base> + $0008_H$ |
| | CSIHn memory control register 0 | CSIHnMCTL0 | <CSIHn_base> + $1040_H$ |
| | CSIHn memory control register 1 | CSIHnMCTL1 | <CSIHn_base> + $1000_H$ |
| | CSIHn memory control register 2 | CSIHnMCTL2 | <CSIHn_base> + $1004_H$ |
| | CSIHn memory read/write pointer register 0 | CSIHnMRWP0 | <CSIHn_base> + $1018_H$ |
| | CSIHn configuration register 0 | CSIHnCFG0 | <CSIHn_base> + $1044_H$ |
| | CSIHn configuration register 1 | CSIHnCFG1 | <CSIHn_base> + $1048_H$ |
| | CSIHn configuration register 2 | CSIHnCFG2 | <CSIHn_base> + $104C_H$ |
| | CSIHn configuration register 3 | CSIHnCFG3 | <CSIHn_base> + $1050_H$ |
| | CSIHn configuration register 4 | CSIHnCFG4 | <CSIHn_base> + $1054_H$ |
| | CSIHn configuration register 5 | CSIHnCFG5 | <CSIHn_base> + $1058_H$ |
| | CSIHn configuration register 6 | CSIHnCFG6 | <CSIHn_base> + $105C_H$ |
| | CSIHn configuration register 7 | CSIHnCFG7 | <CSIHn_base> + $1060_H$ |
| | CSIHn transmit data register 0 for word access | CSIHnTX0W | <CSIHn_base> + $1008_H$ |
| | CSIHn transmit data register 0 for half word access | CSIHnTX0H | <CSIHn_base> + $100C_H$ |
| | CSIHn receive data register 0 for word access | CSIHnRX0W | <CSIHn_base> + $1010_H$ |
| | CSIHn receive data register 0 for half word access | CSIHnRX0H | <CSIHn_base> + $1014_H$ |
| | CSIHn emulation register | CSIHnEMU | <CSIHn_base> + $0018_H$ |
| | CSIHn baud rate setting register 0 | CSIHnBRS0 | <CSIHn_base> + $1068_H$ |
| | CSIHn baud rate setting register 1 | CSIHnBRS1 | <CSIHn_base> + $106C_H$ |
| | CSIHn baud rate setting register 2 | CSIHnBRS2 | <CSIHn_base> + $1070_H$ |
| | CSIHn baud rate setting register 3 | CSIHnBRS3 | <CSIHn_base> + $1074_H$ |

## 17.3.2 CSIHnCTL0 — CSIHn Control Register 0

This register controls the operation clock, enables/disables transmission/reception, and enables/disables the memory allocated for transmission and/or reception. It forces the stop of communication at the end of the current job.

**Access:** This register can be read or written in 8-bit units.

**Address:** <CSIHn_base> + $0000_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CSIHnPWR | CSIHnTXE | CSIHnRXE | — | — | — | CSIHnJOBE | CSIHnMBS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R | R | R | R/W | R/W |

**Table 17.13 CSIHnCTL0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | CSIHnPWR | Controls the operation clock.<br>0: Stops operation clock.<br>1: Supplies operation clock.<br>Clearing CSIHnPWR to 0 resets the internal circuits, stops operation, and sets CSIH to standby state. Clock supply to internal circuits stops.<br><br>If CSIHnPWR is cleared (to 0) during communication, ongoing communication is immediately aborted. In this case, the communication setting must be reconfigured. |
| 6 | CSIHnTXE | Enables/disables transmission.<br>0: Disables transmission.<br>1: Enables transmission. |
| 5 | CSIHnRXE | Enables/disables reception.<br>0:Disables reception.<br>1: Enables reception. |
| 4 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | CSIHnJOBE | Stops communication at the end of the current job (communication ends if data is written to the transmission buffer when CSIHnTX0W.CSIHnEOJ = 1 (job completion)).<br>0: Communication stop is not requested.<br>1: Stops communication.<br>This bit can be used to abort an ongoing job. This bit is cleared to 0 automatically. Even if this bit is set to 1, the read value is always 0.<br>In FIFO mode, the pointer must be cleared by setting CSIHnSTCR0.CSIHnPCT = 1 before the next communication is started |
| 0 | CSIHnMBS | Bypasses the memory for transmission and/or reception data.<br>0: Memory mode<br>CSIH memory is used for transmission and/or reception data.<br>1: Direct access mode<br>CSIH memory is bypassed. |

**CAUTION**

When setting this register, see **Table 17.32, Notes on Setting Registers**.

### 17.3.3 CSIHnCTL1 — CSIHn Control Register 1

This register specifies the interrupt timing and the interrupt delay mode. It also enables/disables extended data length control, data consistency check, loop-back mode, handshake functionality, and job mode. It selects the active output level of each chip select signal and the behavior of the chip select signals after the transfer of the final data.

**Access:** This register can be read or written in 32-bit units.

**Address:** <CSIHn_base> + $0010_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CSIHn SLRS | — | — | — | — | — | CSIHn PHE | CSIHn CKR | CSIHn SLIT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHn CSL7 | CSIHn CSL6 | CSIHn CSL5 | CSIHn CSL4 | CSIHn CSL3 | CSIHn CSL2 | CSIHn CSL1 | CSIHn CSL0 | CSIHn EDLE | CSIHn JE | CSIHn DCS | CSIHn CSRI | CSIHn LBM | CSIHn SIT | CSIHn HSE | CSIHn SSE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 17.14 CSIHnCTL1 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 25 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 24 | CSIHnSLRS | Sets the internal synchronization timing for receive data input.<br>0: Rising edge of PCLK<br>1: Falling edge of PCLK<br>For differences by the setting, see Data Sheet. |
| 23 to 19 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 18 | CSIHnPHE | Sets the CPU-controlled priority-based communication function.<br>0: The CPU-controlled high-priority communication function is disabled.<br>1: The CPU-controlled high-priority communication function is enabled.<br>To enable the CPU-controlled high-priority communication function, set this bit to 1 and set CSIHnJE = 1. This bit can only be set in transmit-only buffer mode. |
| 17 | CSIHnCKR | CSIHTSCK Clock Inversion Function<br>0: The default level of CSIHTSCK is high<br>1: The default level of CSIHTSCK is low<br>For details, see **Section 17.3.11, CSIHnCFGx — CSIHn Configuration Register x**. |
| 16 | CSIHnSLIT | Selects the timing of interrupt INTCSIHTIC.<br>0: Normal interrupt timing (interrupt is generated after the transfer)<br>1: As soon as the contents of the CSIHnTX0W/H register are transferred to the shift register, an interrupt is generated (this function is activated only in direct access mode/transmit-only buffer mode).<br>For details, see **Section 17.4.3, INTCSIHTIC (Communication Status Interrupt)**. |
| 15 to 8 | CSIHnCSLx | Selects the active output level of chip select signal x (CSIHTCSSx).<br>0: Chip select is active low.<br>1: Chip select is active high.<br>For details, see **Section 17.5.3, Chip Selection (CS) Features**. |
| 7 | CSIHnEDLE | Enables/disables extended data length (EDL) mode.<br>0: Disables extended data length mode.<br>1: Enables extended data length mode.<br>For details, see **Section 17.5.8.2, Data Length Greater than 16 Bits**. |

**Table 17.14 CSIHnCTL1 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 6 | CSIHnJE | Enables/disables job mode.<br>0: Disables job mode.<br>1: Enables job mode.<br>For details, see **Section 17.5.3.3, Job Concept**.<br>The CSIHnCTL0.CSIHnJOBE, CSIHnTX0W.CSIHnEOJ, and CSIHnTX0W.CSIHnCIRE bits are enabled only when CSIHnJE = 1.<br>Setting this bit in slave mode is prohibited.<br>In addition, to enable the CPU-controlled high-priority communication function, set this bit to 1 as well as CSIHnPHE = 1. |
| 5 | CSIHnDCS | Enables/disables data consistency check.<br>0: Disables data consistency check.<br>1: Enables data consistency check.<br>For details, see **Section 17.5.12.1, Data Consistency Check**. |
| 4 | CSIHnCSRI | Defines chip select signal behavior after last data transfer.<br>0: Chip select signal retains the active level.<br>1: Chip select signal returns to the inactive level.<br>The last data is determined at the interrupt timing in direct access mode or FIFO mode. When CSIHnCTL1.CSIHnSLIT = 1, the last data is determined in direct access mode. |
| 3 | CSIHnLBM | Controls loop-back mode (LBM).<br>0: Deactivates loop-back mode.<br>1: Activates loop-back mode.<br>For details, see **Section 17.5.13, Loop-Back Mode**. |
| 2 | CSIHnSIT | Selects interrupt delay mode.<br>0: No delay is generated.<br>1: Half clock delay is generated for all interrupts.<br>This bit is only valid in master mode. In slave mode, no delay is generated.<br>For details, see **Section 17.4.2, Interrupt Delay**. |
| 1 | CSIHnHSE | Enables/disables the handshake function.<br>0: Disables the handshake function.<br>1: Enables the handshake function.<br>For details see **Section 17.5.11, Handshake Function**. |
| 0 | CSIHnSSE | Enables/disables the slave select function.<br>0: Input signal $\overline{\text{CSIHTSSI}}$ is disabled.<br>1: Input signal $\overline{\text{CSIHTSSI}}$ is recognized.<br>If the slave select function is not used, this bit must be set to 0 (see also **Section 17.5.2, Master/Slave Connections**). |

Details about CSIHnCTL1.CSIHnSSE are shown in the following tables.

**Table 17.15 Operation of the Slave Select Function during Reception**

| CSIHnCTL0. CSIHnRXE | CSIHnCTL1. CSIHnSSE | $\overline{\text{CSIHTSSI}}$ | Receive Operation |
|---|---|---|---|
| 0 | — | — | Reception is prohibited |
| 1 | 0 | — | Possible |
| 1 | 1 | 0 | Possible |
| 1 | 1 | 1 | Disabled |

**Table 17.16 Operation of the Slave Select Function during Transmission**

| CSIHnCTL0. CSIHnTXE | CSIHnCTL1. CSIHnSSE | $\overline{\text{CSIHTSSI}}$ | Transmit Operation |
|---|---|---|---|
| 0 | — | — | Transmission is prohibited |
| 1 | 0 | — | Possible |
| 1 | 1 | 0 | Possible |
| 1 | 1 | 1 | Disabled |

**CAUTION**

When setting this register, see **Table 17.32, Notes on Setting Registers**.

### 17.3.4 CSIHnCTL2 — CSIHn Control Register 2

This register selects operating mode and the reference clock value, and specifies the transfer clock frequency.

For details see **Section 17.5.5, Transmission Clock Selection**.

**Access:** This register can be read or written in 16-bit units.

**Address:** <CSIHn_base> + $0014_H$

**Value after reset:** $E000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnPRS[2:0] | | | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 17.17 CSIHnCTL2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 13 | CSIHnPRS[2:0] | These bits select the operation mode and the reference clock value. |
| 12 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

| CSIHnPRS2 | CSIHnPRS1 | CSIHnPRS0 | Selection of Reference Clock (PRSOUT) |
|---|---|---|---|
| 0 | 0 | 0 | PCLK (Master mode) |
| 0 | 0 | 1 | PCLK/2 (Master mode) |
| 0 | 1 | 0 | PCLK/4 (Master mode) |
| 0 | 1 | 1 | PCLK/8 (Master mode) |
| 1 | 0 | 0 | PCLK/16 (Master mode) |
| 1 | 0 | 1 | PCLK/32 (Master mode) |
| 1 | 1 | 0 | PCLK/64 (Master mode) |
| 1 | 1 | 1 | External clock via CSIHTSCK(in) (Slave mode) |

In master mode, the following bits are used to set the transfer clock frequency:

CSIHnCTL2.CSIHnPRS[2:0], CSIHnCFGx.CSIHnBRSS[1:0], CSIHnBRSy.CSIHnBRS[11:0]

In addition, any of the four different transfer clock frequency settings that are specified by the CSIHnBRSy.CSIHnBRS[11:0] bits is selected for each chip select signal. To select the transfer clock frequency setting for each chip select signal, use the CSIHnCFGx.CSIHnBRSS[1:0] bits.

The following table shows the relationship between CSIHnCFGx.CSIHnBRSS[1:0] and CSIHnBRSy.CSIHnBRS[11:0].

| CSIHnCFGx. CSIHnBRSS[1:0] | Transfer Clock Frequency Setting Bit to be Selected |
|---|---|
| 00 | CSIHnBRS0.CSIHnBRS[11:0] |
| 01 | CSIHnBRS1.CSIHnBRS[11:0] |
| 10 | CSIHnBRS2.CSIHnBRS[11:0] |
| 11 | CSIHnBRS3.CSIHnBRS[11:0] |

The following table shows the relationship between the transfer clock frequency and the transfer clock frequency setting (CSIHnBRSy[11:0]) selected by the CSIHnBRSS[1:0] bits when the bit value of the CSIHnPRS[2:0] bits is α.

| CSIHnBRSy[11:0] | Transfer clock frequency |
|---|---|
| 0 | BRG stopped |
| 1 | PCLK / ($2^{\alpha}$ × 1 × 2) |
| 2 | PCLK / ($2^{\alpha}$ × 2 × 2) |
| 3 | PCLK / ($2^{\alpha}$ × 3 × 2) |
| 4 | PCLK / ($2^{\alpha}$ × 4 × 2) |
| ... | … |
| 4095 | PCLK / ($2^{\alpha}$ × 4095× 2) |

When a time-out error is used in slave mode, the clock selected by this setting is used. In slave mode, the CSIHnPRS[2:0] bits are set to $111_B$. In this case, the prescaler has the same setting as when the CSIHnPRS[2:0] bits are set to $000_B$. When using a time-out error, set the CSIHnBRSy.CSIHnBRS[11:0] bits to a value other than $000_H$.

**CAUTION**

When setting this register, see **Table 17.32, Notes on Setting Registers**.

### 17.3.5 CSIHnSTR0 — CSIHn Status Register 0

This register indicates the status of CSIH.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <CSIHn_base> + $0004_H$

**Value after reset:** $0000\ 0010_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnSRP[7:0] | | | | | | | | CSIHnSPF[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHn TMOE | CSIHn OFE | — | — | — | — | — | CSIHn HPST | CSIHn TSF | — | CSIHn FLF | CSIHn EMF | CSIHn DCE | — | CSIHn PE | CSIHn OVE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 17.18 CSIHnSTR0 Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | CSIHnSRP[7:0] | Indicates the number of received data in FIFO mode.<br>**CSIHnSRP[7:0]** / **Description**<br>$00_H$ / Number of received data packets (0 to 128)<br>…<br>$80_H$<br>Other than the above / Undefined<br>These bits are cleared by CSIHnSTCR0.CSIHnPCT.<br>In direct access mode, dual buffer mode, or transmit-only buffer mode, this value is fixed to $00_H$.<br>In direct access mode, this bit is fixed to 0 because there is no pointer. In buffer mode, this bit is fixed to 0 because the number of data is managed by CSIHnMCTL2.CSIHnND[7:0]. |
| 23 to 16 | CSIHnSPF[7:0] | Indicates the number of unsent data in FIFO mode.<br>(The number of data written by the CPU is the number of sent data.)<br>**CSIHnSPF[7:0]** / **Description**<br>$00_H$ / Number of unsent data (0 to 128)<br>…<br>$80_H$<br>Other than the above / Undefined<br>These bits are cleared by CSIHnSTCR0.CSIHnPCT.<br>In direct access mode, dual buffer mode, or transmit-only buffer mode, this value is fixed to $00_H$.<br>In direct access mode, this bit is fixed to 0 because there is no pointer. In buffer mode, this bit is fixed to 0 because the number of data is managed by CSIHnMCTL2.CSIHnND[7:0]. |

**Table 17.18 CSIHnSTR0 Register Contents (2/3)**

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15</td><td>CSIHnTMOE</td><td>Time-out Error Flag in FIFO Mode<br>Indicates whether a time-out error was detected in FIFO mode.<br>0: No time-out error was detected.<br>1: A time-out error was detected.<br>For details, see Section 17.5.12.3, Time-Out Error.<br>This bit is cleared by CSIHnSTCR0.CSIHnTMOEC.<br>When setting to 1 by time-out error detection and clearing to 0 by CSIHnSTCR0.CSIHnTMOEC occur simultaneously, setting to 1 takes precedence over clearing to 0.<br>This bit is also initialized when CSIHnCTL0.CSIHnPWR is changed from 0 to 1 or from 1 to 0.</td></tr>
<tr><td>14</td><td>CSIHnOFE</td><td>Overflow Error Flag in FIFO mode<br>Indicates whether an overflow error was detected in FIFO mode.<br>0: No overflow error was detected.<br>1: An overflow error was detected.<br>For details, see Section 17.5.12.4, Overflow Error.<br>This bit is cleared by CSIHnSTCR0.CSIHnOFEC.<br>When setting to 1 by overflow error detection and clearing to 0 by CSIHnSTCR0.CSIHnOFEC occur simultaneously, setting to 1 takes precedence over clearing to 0.<br>This bit is also initialized when CSIHnCTL0.CSIHnPWR is changed from 0 to 1 or from 1 to 0.</td></tr>
<tr><td>13 to 9</td><td>Reserved</td><td>When read, the value after reset is returned.</td></tr>
<tr><td>8</td><td>CSIHnHPST</td><td>Communication Priority Indication Flag<br>0: Indicates low-priority communication is in progress.<br>1: Indicates high-priority communication is in progress.<br>This bit always reads 0 if CPU-controlled high-priority communication is disabled (CSIHnCTL1.CSIHnPHE = 0).</td></tr>
<tr><td>7</td><td>CSIHnTSF</td><td>Transfer Status Flag<br>0: Idle state<br>1: Communication is in progress or being prepared.<br>The timing to set or clear this bit is as follows:
<table>
<tr><th rowspan="2">Master Mode</th><th colspan="2">Timing to Set</th><th rowspan="2">Timing to Clear</th></tr>
<tr><th>Direct Access Mode,<br>FIFO Mode</th><th>Dual Buffer Mode, Transmit-Only Buffer Mode</th></tr>
<tr><td>Transmit-only mode</td><td rowspan="3">Data is written to a transmit register (CSIHnTX0W/ CSIHnTX0H)</td><td rowspan="3">CSIHnMCTL2.CSIHnBTST bit is set</td><td rowspan="3">Within a half clock cycle the last serial clock edge</td></tr>
<tr><td>Transmit/receive mode</td></tr>
<tr><td>Receive-only mode</td></tr>
</table>
<table>
<tr><th rowspan="2">Slave Mode</th><th colspan="2">Timing to Set</th><th rowspan="2">Timing to Clear</th></tr>
<tr><th>Direct Access Mode,<br>FIFO Mode</th><th>Dual Buffer Mode, Transmit-Only Buffer Mode</th></tr>
<tr><td>Transmit-only mode</td><td rowspan="2">Data is written to a transmit register (CSIHnTX0W/ CSIHnTX0H)</td><td rowspan="3">CSIHnMCTL2.CSIHnBTST bit is set</td><td rowspan="3">Within a half clock cycle the last serial clock edge</td></tr>
<tr><td>Transmit/receive mode</td></tr>
<tr><td>Receive-only mode</td><td>Input timing of CSIHTSCK</td></tr>
</table></td></tr>
<tr><td>6</td><td>Reserved</td><td>When read, the value after reset is returned.</td></tr>
<tr><td>5</td><td>CSIHnFLF</td><td>A flag indicating that the buffer is full in FIFO mode.<br>0: FIFO buffer is not full.<br>1: FIFO buffer is full.<br>This bit is cleared by CSIHnSTCR0.CSIHnPCT.<br>The FIFO buffer might be filled with unsent data or received data.</td></tr>
</table>

**Table 17.18 CSIHnSTR0 Register Contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 4 | CSIHnEMF | A flag indicating that the buffer is empty in FIFO mode.<br>0: FIFO buffer is not empty.<br>1: FIFO buffer is empty.<br>This bit is set to 1 by CSIHnSTCR0.CSIHnPCT.<br>This bit is set to 1 when CSIHnSTR0.CSIHnSRP[7:0] + CSIHnSTR0.CSIHnSPF[7:0] = $00_H$.<br>The FIFO buffer might be filled with unsent data or received data. |
| 3 | CSIHnDCE | Data Consistency Check Error Flag<br>0: No data consistency check error is detected.<br>1: Data consistency check error is detected.<br>This bit is cleared by writing 1 to CSIHnSTCR0.CSIHnDCEC.<br>When setting to 1 by data consistency check error detection and clearing to 0 by CSIHnSTCR0.CSIHnDCEC occur simultaneously, setting to 1 takes precedence over clearing to 0.<br>This bit is initialized when CSIHnCTL0.CSIHnPWR is changed from 0 to 1 or from 1 to 0. |
| 2 | Reserved | When read, the value after reset is returned. |
| 1 | CSIHnPE | Parity Error Flag<br>0: No parity error is detected.<br>1: Parity error is detected.<br>This bit is cleared by writing 1 to CSIHnSTCR0.CSIHnPEC.<br>When setting to 1 due to parity error detection and clearing to 0 by CSIHnSTCR0.CSIHnPEC occur simultaneously, setting to 1 by parity error detection takes precedence over clearing to 0.<br>This bit is initialized when CSIHnCTL0.CSIHnPWR changes from 0 to 1 or from 1 to 0. |
| 0 | CSIHnOVE | Overrun Error Flag (Fixed to 0 in dual buffer mode)<br>0: No overrun error is detected.<br>1: Overrun error is detected.<br>This bit is cleared by writing 1 to CSIHnSTCR0.CSIHnOVEC. When setting to 1 due to overrun error detection and clearing to 0 by writing to CSIHnSTCR0.CSIHnOVEC occur simultaneously, setting to 1 by overrun error detection takes precedence over clearing to 0.<br>This bit is initialized when CSIHnCTL0.CSIHnPWR changes from 0 to 1 or from 1 to 0. |

Table 17.19 Behaviors in Various Memory Modes

| Bit Name | Bit Position | Direct Access Mode | FIFO Mode | Transmit-Only Buffer Mode | Dual Buffer Mode |
|---|---|---|---|---|---|
| CSIHnSRP[7:0] | 31 to 24 | Fixed to 0 | Number of received data | Fixed to 0 | Fixed to 0 |
| CSIHnSPF[7:0] | 23 to 16 | Fixed to 0 | Number of untransmitted data packets | Fixed to 0 | Fixed to 0 |
| CSIHnTMOE | 15 | Fixed to 0 | 0: No error is detected.<br>1: An error is detected. | Fixed to 0 | Fixed to 0 |
| CSIHnOFE | 14 | Fixed to 0 | 0: No error is detected.<br>1: An error is detected. | Fixed to 0 | Fixed to 0 |
| CSIHnTSF | 7 | 0: Idle state<br>1: Communication is in progress or being prepared | | | |
| CSIHnFLF | 5 | Fixed to 0 | 0: FIFO is not full<br>1: FIFO is full | Fixed to 0 | Fixed to 0 |
| CSIHnEMF | 4 | Fixed to 1 | 0: FIFO is not empty<br>1: FIFO is empty | Fixed to 1 | Fixed to 1 |
| CSIHnDCE | 3 | 0: No error is detected.<br>1: An error is detected. | | | |
| CSIHnPE | 1 | 0: No error is detected.<br>1: An error is detected. | | | |
| CSIHnOVE | 0 | 0: No error is detected.<br>1: An error is detected. | 0: No error is detected.<br>1: An error is detected. | 0: No error is detected.<br>1: An error is detected. | Fixed to 0 |

**CAUTION**

When setting this register, see **Table 17.32, Notes on Setting Registers**.

### 17.3.6 CSIHnSTCR0 — CSIHn Status Clear Register 0

This register clears the status flags of the CSIHnSTR0 status register.

**Access:** This register can be read or written in 16-bit units.
When read, the value $0000_H$ is always returned.

**Address:** <CSIHn_base> + $0008_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHn TMOEC | CSIHn OFEC | — | — | — | — | — | CSIHn PCT | — | — | — | — | CSIHn DCEC | — | CSIHn PEC | CSIHn OVEC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R/W | R | R | R | R | R/W | R | R/W | R/W |

**Table 17.20 CSIHnSTCR0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | CSIHnTMOEC | Controls the time-out error flag clear command.<br>0: No operation. The read value is always 0.<br>1: Clears the time-out error flag (CSIHnSTR0.CSIHnTMOE). |
| 14 | CSIHnOFEC | Controls the overflow error flag clear command.<br>0: No operation. The read value is always 0.<br>1: Clears the overflow error flag (CSIHnSTR0.CSIHnOFE). |
| 13 to 9 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 8 | CSIHnPCT | Controls the FIFO pointer clear command.<br>0: No operation. The read value is always 0.<br>1: Clears the following FIFO buffer pointers (in FIFO mode, dual buffer mode, and transmit-only buffer mode) and status bits.<br>**FIFO Buffer Pointer / Status Bit**<br>CSIHnMRWP0.CSIHnTRWA[6:0] — CSIHnSTR0.CSIHnSPF[7:0]<br>CSIHnMRWP0.CSIHnRRA[6:0] — CSIHnSTR0.CSIHnSRP[7:0]<br>CSIHnMCTL2.CSIHnSOP[6:0] — CSIHnSTR0.CSIHnFLF, CSIHnSTR0.CSIHnTSF<br>Additionally, the CSIHnSTR0.CSIHnEMF bit is set to 1 (FIFO empty) (in FIFO mode only). |
| 7 to 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 | CSIHnDCEC | Controls the data consistency check error flag clear command.<br>0: No operation. The read value is always 0.<br>1: Clears the data consistency check error flag (CSIHnSTR0.CSIHnDCE). |
| 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | CSIHnPEC | Controls the parity error flag clear command.<br>0: No operation. The read value is always 0.<br>1: Clears the parity error flag (CSIHnSTR0.CSIHnPE). |
| 0 | CSIHnOVEC | Controls the overrun error flag clear command.<br>0: No operation. The read value is always 0.<br>1: Clears the overrun error flag (CSIHnSTR0.CSIHnOVE). |

**CAUTION**

When setting this register, see **Table 17.32, Notes on Setting Registers**.

## 17.3.7 CSIHnMCTL0 — CSIHn Memory Control Register 0

This register selects the memory mode and the time-out setting.

**Access:** This register can be read or written in 16-bit units.

**Address:** <CSIHn_base> + $1040_H$

**Value after reset:** $001F_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CSIHn MMS[1:0] | | — | — | — | CSIHnTO[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 17.21 CSIHnMCTL0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 10 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 9 to 8 | CSIHnMMS [1:0] | Selects the memory mode.<br>**CSIHn MMS1 / CSIHn MMS0 / Description**<br>0 / 0 / FIFO mode<br>0 / 1 / Dual buffer mode<br>1 / 0 / Transmit-only buffer mode<br>1 / 1 / Prohibited<br>After a change of the memory mode, the respective buffer pointers must be cleared by setting the CSIHnSTCR0.CSIHnPCT bit to 1.<br>In direct access mode, the setting of these bits is ignored. |
| 7 to 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 0 | CSIHnTO[4:0] | Selects the time-out setting in FIFO mode.<br>**CSIHnTO[4:0] / Description**<br>$00000_B$ / No time-out is detected<br>$00001_B$ / Time-out is (1 × 8 × BRG output clocks)<br>$00010_B$ / Time-out is (2 × 8 × BRG output clocks)<br>...<br>$11111_B$ / Time-out is (31 × 8 × BRG output clocks)<br>**CAUTION**<br>Changing the time-out setting is only permitted when CSIHnCTL0.CSIHnPWR = 0.<br>Set the CSIHnTO[4:0] bits to $00000_B$ in direct access mode, dual buffer mode, or transmit-only buffer mode (except FIFO mode).<br>For details about time-out detection, see also **Section 17.5.12.3, Time-Out Error**. |

**CAUTION**

When setting this register, see **Table 17.32, Notes on Setting Registers**.

### 17.3.8 CSIHnMCTL1 — CSIHn Memory Control Register 1

This register selects the conditions to generate the interrupt requests INTCSIHTIC and INTCSIHTIR in FIFO mode.

**Access:** This register can be read or written in 32-bit units.

**Address:** <CSIHn_base> + $1000_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CSIHnFES[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CSIHnFFS[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 17.22 CSIHnMCTL1 Register Contents**

| Bit Position | Bit name | Function |
|---|---|---|
| 31 to 23 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 22 to 16 | CSIHnFES[6:0] | Selects the conditions to generate the INTCSIHTIC interrupt (transmit data empty) in FIFO mode.<br>When the number of unsent data to be transmitted in FIFO (checked by the CSIHnSTR0.CSIHnSPF[7:0] bit) and CSIHnMCTL1.CSIHnFES[6:0] match, the FIFO empty flag (CSIHnSTR0.CSIHnEMF bit) is set to 1, and the INTCSIHTIC interrupt request is generated. |
| 15 to 7 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 6 to 0 | CSIHnFFS[6:0] | Selects the conditions to generate the INTCSIHTIR interrupt (receive data full) in FIFO mode.<br>When the number of received data in FIFO (checked by the CSIHnSTR0.CSIHnSRP[7:0] bit) and (128 - CSIHnMCTL1.CSIHnFFS[6:0]) match, the INTCSIHTIR interrupt request is generated. |

**CAUTION**

When setting this register, see **Table 17.32, Notes on Setting Registers**.

### 17.3.9 CSIHnMCTL2 — CSIHn Memory Control Register 2

This register controls the operation of the memory in dual buffer or transmit-only buffer mode and triggers to start communication.

**Access:** This register can be read or written in 32-bit units.

**Address:** <CSIHn_base> + $1004_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHn BTST | — | — | — | — | — | — | — | CSIHnND[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CSIHnSOP[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 17.23 CSIHnMCTL2 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | CSIHnBTST | Provides a start trigger for buffer transfer.<br>0: No operation.<br>1: Issues the start transfer command.<br>The read value is always 0.<br>**CAUTION**<br>This bit can only be used in dual buffer mode and transmit-only buffer mode. |
| 30 to 24 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 23 to 16 | CSIHnND[7:0] | Specifies the number of data for each memory mode.<br>The read value indicates the number of remaining communication data. |

| CSIHn ND[7:0] | Dual Buffer Mode | Transmit-Only Buffer Mode | FIFO Mode | Direct Access Mode |
|---|---|---|---|---|
| $00_H$ | Send 0 data | Send 0 data | No influence | No influence |
| $01_H$ | Send 1 data | Send 1 data | No influence | No influence |
| ... | ... | ... | No influence | No influence |
| $3F_H$ | Send 63 data | Send 63 data | No influence | No influence |
| $40_H$ | Send 64 data | Send 64 data | No influence | No influence |
| ... | Prohibited | ... | No influence | No influence |
| $7F_H$ | Prohibited | Send 127 data | No influence | No influence |
| $80_H$ | Prohibited | Send 128 data | No influence | No influence |
| Other than the above | Setting is prohibited. | | | |

The values are automatically decremented after data transfer (not decremented in direct access mode).

| | | |
|---|---|---|
| 15 to 7 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

**Table 17.23 CSIHnMCTL2 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 6 to 0 | CSIHnSOP[6:0] | Selects the pointer of the data to be sent.<br>If communication is forced to stop by setting CSIHnCTL0.CSIHnPWR to 0 or CSIHnSTCR0.CSIHnPCT to 1, these bits are cleared by hardware.<br>In FIFO mode, these bits indicate the send address. |

| CSIHn SOP[6:0] | Dual Buffer Mode | Transmit-Only Buffer Mode | FIFO Mode | Direct Access Mode |
|---|---|---|---|---|
| $00_H$ | $0000_H$ | $0000_H$ | $0000_H$ | No influence |
| $01_H$ | $0004_H$ | $0004_H$ | $0004_H$ | No influence |
| ... | ... | ... | ... | No influence |
| $3F_H$ | $00FC_H$ | $00FC_H$ | $00FC_H$ | No influence |
| $40_H$ | Prohibited | $0100_H$ | $0100_H$ | No influence |
| ... | Prohibited | ... | ... | No influence |
| $7F_H$ | Prohibited | $01FC_H$ | $01FC_H$ | No influence |

**CAUTION**

In direct access mode, these bits are not incremented.

**CAUTION**

When setting this register, see **Table 17.32, Notes on Setting Registers**.

## 17.3.10 CSIHnMRWP0 — CSIHn Memory Read/Write Pointer Register 0

This register sets the pointers for reading from and writing to the dual buffer or transmit-only buffer.

**Access:** This register can be read or written in 32-bit units.

**Address:** <CSIHn_base> + $1018_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CSIHnRRA[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CSIHnTRWA[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 17.24 CSIHnMRWP0 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 22 to 16 | CSIHnRRA[6:0] | Selects the read pointer of the receive buffer. |

| CSIHn RRA[6:0] | Dual Buffer Mode | Transmit-Only Buffer Mode | FIFO Mode | Direct Access Mode |
|---|---|---|---|---|
| $00_H$ | $0000_H$ | No influence | $0000_H$ | No influence |
| $01_H$ | $0004_H$ | No influence | $0004_H$ | No influence |
| ... | ... | No influence | ... | No influence |
| $3F_H$ | $00FC_H$ | No influence | $00FC_H$ | No influence |
| $40_H$ | Prohibited | No influence | $0100_H$ | No influence |
| ... | Prohibited | No influence | ... | No influence |
| $7F_H$ | Prohibited | No influence | $01FC_H$ | No influence |

These bits are automatically incremented when received data is read.
If an overrun error is generated while reading the CSIHnRX0W or CSIHnRX0H register, the read pointer is not incremented.
These bits are cleared when CSIHnSTCR0.CSIHnPCT is set to 1.
In direct access mode and transmit-only buffer mode, these bits are not incremented.
To perform write access in transmit-only buffer mode, set $00_H$ to these bits.
In FIFO mode, these bits indicate the read address of the received data.

| | | |
|---|---|---|
| 15 to 7 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

**Table 17.24 CSIHnMRWP0 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 6 to 0 | CSIHnTRWA [6:0] | Selects the read/write pointer of the transmit buffer. |

| CSIHn TRWA[6:0] | Dual Buffer Mode | Transmit-Only Buffer Mode | FIFO Mode | Direct Access Mode |
|---|---|---|---|---|
| $00_H$ | $0000_H$ | $0000_H$ | $0000_H$ | No influence |
| $01_H$ | $0004_H$ | $0004_H$ | $0004_H$ | No influence |
| ... | ... | ... | ... | No influence |
| $3F_H$ | $00FC_H$ | $00FC_H$ | $00FC_H$ | No influence |
| $40_H$ | Prohibited | $0100_H$ | $0100_H$ | No influence |
| ... | Prohibited | ... | ... | No influence |
| $7F_H$ | Prohibited | $01FC_H$ | $01FC_H$ | No influence |

These bits are automatically incremented when the transmission data is written or read.
These bits are cleared when CSIHnSTCR0.CSIHnPCT is set to 1.
In direct access mode, these bits are not incremented.
In FIFO mode, these bits indicate the read/write address of transmission data.

**CAUTION**

When setting this register, see **Table 17.32, Notes on Setting Registers**.

## 17.3.11 CSIHnCFGx — CSIHn Configuration Register x

These eight registers configure the prescaler, parity, data length, recessive configuration for broadcasting, serial data direction, clock phase and data phase, idle enforcement configuration, idle time, hold time, inter-data time, and setup for each chip select signal, CSIHTCSSx.

### Slave mode

In slave mode, the transmission protocol setting of the CSIHnCFG0 register is effective.

- CSIHnPSx[1:0]: parity usage
- CSIHnDLSx[3:0]: data length selection
- CSIHnDIRx: data direction
- CSIHnCKPx, CSIHnDAPx: clock phase and data phase

In slave mode, set all bits other than above in the CSIHnCFG0 register, and the CSIHnCFG1 to CSIHnCFG7 registers to 0.

**Access:** This register can be read or written in 32-bit units.

**Address:** CSIHnCFG0: <CSIHn_base> + $1044_H$
CSIHnCFG1: <CSIHn_base> + $1048_H$
CSIHnCFG2: <CSIHn_base> + $104C_H$
CSIHnCFG3: <CSIHn_base> + $1050_H$
CSIHnCFG4: <CSIHn_base> + $1054_H$
CSIHnCFG5: <CSIHn_base> + $1058_H$
CSIHnCFG6: <CSIHn_base> + $105C_H$
CSIHnCFG7: <CSIHn_base> + $1060_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnBRSSx[1:0] | | CSIHnPSx[1:0] | | CSIHnDLSx[3:0] | | | | — | — | — | — | CSIHnRCBx | CSIHnDIRx | CSIHnCKPx | CSIHnDAPx |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnIDLx | CSIHnIDx[2:0] | | | CSIHnHDx[3:0] | | | | CSIHnINx[3:0] | | | | CSIHnSPx[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 17.25 CSIHnCFGx Register Contents (1/4)**

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>31, 30</td><td>CSIHnBRSSx [1:0]</td><td>These bits select the baud rate setting register (CSIHnBRSy).
<table>
<tr><th>CSIHn BRSSx1</th><th>CSIHn BRSSx0</th><th>Baud Rate Setting Register Selection</th></tr>
<tr><td>0</td><td>0</td><td>The transfer clock frequency is set according to the CSIHnBRS0 setting.</td></tr>
<tr><td>0</td><td>1</td><td>The transfer clock frequency is set according to the CSIHnBRS1 setting.</td></tr>
<tr><td>1</td><td>0</td><td>The transfer clock frequency is set according to the CSIHnBRS2 setting.</td></tr>
<tr><td>1</td><td>1</td><td>The transfer clock frequency is set according to the CSIHnBRS3 setting.</td></tr>
</table>
The maximum value for setting the transfer clock frequency, in accordance with the CSIHnCTL2.CSIHnPRS[2:0] setting, must be as follows:<br>Master mode: PCLK/4<br>Slave mode: PCLK/16</td></tr>
<tr><td>29, 28</td><td>CSIHnPSx[1:0]</td><td>Selects the parity for transmitting or receiving chip select signal x.
<table>
<tr><th>CSIHn PSx1</th><th>CSIHn PSx0</th><th>Transmission</th><th>Reception</th></tr>
<tr><td>0</td><td>0</td><td>Does not transmit any parity bit.</td><td>Does not wait for reception of the parity bit.</td></tr>
<tr><td>0</td><td>1</td><td>Adds a parity bit fixed to 0.</td><td>Waits for reception of the parity bit but does not evaluate it.</td></tr>
<tr><td>1</td><td>0</td><td>Adds the odd parity bit.</td><td>Waits for the odd parity bit.</td></tr>
<tr><td>1</td><td>1</td><td>Adds the even parity bit.</td><td>Waits for the even parity bit.</td></tr>
</table></td></tr>
<tr><td>27 to 24</td><td>CSIHnDLSx [3:0]</td><td>Selects the data length for chip select signal x.
<table>
<tr><th>CSIHnDLSx[3:0]</th><th>Data Length</th></tr>
<tr><td>$0000_B$</td><td>16 bits</td></tr>
<tr><td>$0001_B$</td><td>1 bit</td></tr>
<tr><td>$0010_B$</td><td>2 bits</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>$1111_B$</td><td>15 bits</td></tr>
</table>
CAUTION<br>When CSIHnTX0W.CSIHnEDL = 1, the setting of this bit has no effect. When CSIHnTX0W.CSIHnEDL = 0 (the data length is 16 bits), the setting of this bit is valid. Setting “1 bit” is only enabled if the previous transmit data was 16 bits with CSIHnEDL = 1.</td></tr>
<tr><td>23 to 20</td><td>Reserved</td><td>When read, the value after reset is returned. When writing, write the value after reset.</td></tr>
<tr><td>19</td><td>CSIHnRCBx</td><td>Selects the recessive configuration for broadcasting for chip select signal x.<br>0: Dominant (higher priority)<br>1: Recessive (lower priority)<br>For details, see Section 17.5.3.1, Configuration Registers</td></tr>
<tr><td>18</td><td>CSIHnDIRx</td><td>Selects the serial data direction of chip select signal x.<br>0: Data is transmitted/received with MSB first.<br>1: Data is transmitted/received with LSB first.<br>For details, see Section 17.5.9, Serial Data Direction Selection.</td></tr>
</table>

**Table 17.25 CSIHnCFGx Register Contents (2/4)**

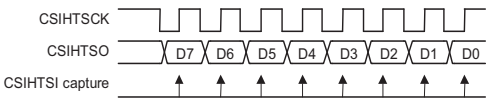
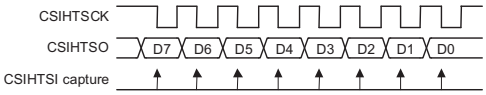
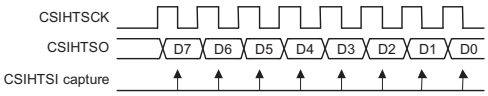
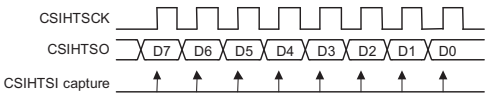
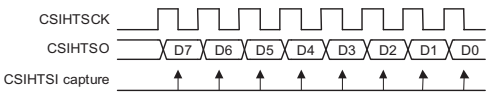
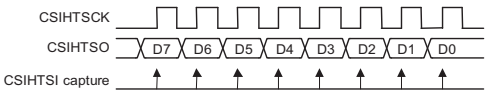
<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>17</td><td>CSIHnCKPx</td><td rowspan="2">CSIHnCKPx: Clock phase selection bit<br>CSIHnDAPx: Data phase selection bit<br>• CSIHnCTL1.CSIHnCKR = 0<br><table><tr><th>CSIHn CKPx</th><th>CSIHn DAPx</th><th>Clock Phase and Data Phase Selection</th></tr><tr><td>0</td><td>0</td><td>  </td></tr><tr><td>0</td><td>1</td><td>  </td></tr><tr><td>1</td><td>0</td><td>  </td></tr><tr><td>1</td><td>1</td><td>  </td></tr></table>• CSIHnCTL1.CSIHnCKR = 1<br><table><tr><th>CSIHn CKPx</th><th>CSIHn DAPx</th><th>Clock Phase and Data Phase Selection</th></tr><tr><td>0</td><td>0</td><td>  </td></tr><tr><td>0</td><td>1</td><td>  </td></tr><tr><td>1</td><td>—</td><td>Setting prohibited</td></tr></table></td></tr>
<tr><td>16</td><td>CSIHnDAPx</td></tr>
<tr><td>15</td><td>CSIHnIDLx</td><td>Selects the idle enforcement configuration for chip select signal x<br>0: If the CSIHnTX0W.CSIHnCSx settings of two consecutive transfers are different, an idle state is inserted between two transfers. If the CSIHnTX0W.CSIHnCSx settings of two consecutive transfers are the same, an idle state is not inserted between two transfers.<br>1: Regardless of the CSIHnTX0W.CSIHnCSx settings of two consecutive transfers, an idle state are inserted between two transfers.<br>This bit is only available in master mode.<br>For details about the enforced idle state, see Section 17.5.15, Enforced Chip Select Idle Setting.</td></tr>
</table>

**Table 17.25 CSIHnCFGx Register Contents (3/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 14 to 12 | CSIHnIDx[2:0] | Selects the idle time for chip select signal x.<br>**CSIHn IDx[2:0]** / **Idle time**<br>$000_B$ 0.5 transmission clock cycle<br>$001_B$ 1.0 transmission clock cycle<br>$010_B$ 1.5 transmission clock cycles<br>$011_B$ 2.5 transmission clock cycles<br>$100_B$ 3.5 transmission clock cycles<br>$101_B$ 4.5 transmission clock cycles<br>$110_B$ 6.5 transmission clock cycles<br>$111_B$ 8.5 transmission clock cycles<br>These bits are only available in master mode. |
| 11 to 8 | CSIHnHDx[3:0] | Specifies the hold time for chip select signal x in transmission clock cycles.<br>**CSIHn HDx[3:0]** / **Hold time when CSIHnCTL1.CSIHnSIT is 0** / **Hold time when CSIHnCTL1.CSIHnSIT is 1**<br>$0000_B$ 0.5 transmission clock cycle 1.0 transmission clock cycle<br>$0001_B$ 1.0 transmission clock cycle 1.5 transmission clock cycles<br>$0010_B$ 1.5 transmission clock cycles 2.0 transmission clock cycles<br>$0011_B$ 2.5 transmission clock cycles 3.0 transmission clock cycles<br>$0100_B$ 3.5 transmission clock cycles 4.0 transmission clock cycles<br>$0101_B$ 4.5 transmission clock cycles 5.0 transmission clock cycles<br>$0110_B$ 6.5 transmission clock cycles 7.0 transmission clock cycles<br>$0111_B$ 8.5 transmission clock cycles 9.0 transmission clock cycles<br>$1000_B$ 9.5 transmission clock cycles 10.0 transmission clock cycles<br>$1001_B$ 10.5 transmission clock cycles 11.0 transmission clock cycles<br>$1010_B$ 11.5 transmission clock cycles 12.0 transmission clock cycles<br>$1011_B$ 12.5 transmission clock cycles 13.0 transmission clock cycles<br>$1100_B$ 14.5 transmission clock cycles 15.0 transmission clock cycles<br>$1101_B$ 16.5 transmission clock cycles 17.0 transmission clock cycles<br>$1110_B$ 18.5 transmission clock cycles 19.0 transmission clock cycles<br>$1111_B$ 20.5 transmission clock cycles 21.0 transmission clock cycles<br>These bits are only available in master mode. |

**Table 17.25 CSIHnCFGx Register Contents (4/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | CSIHnINx[3:0] | Specifies the inter-data time for chip select signal x in transmission clock cycles. |

| CSIHnINx[3:0] | Inter-Data Time when CSIHnCTL1.CSIHnSIT is 0 | Inter-Data Time when CSIHnCTL1.CSIHnSIT is 1 |
|---|---|---|
| $0000_B$ | 0.0 transmission clock cycle | 0.5 transmission clock cycle |
| $0001_B$ | 0.5 transmission clock cycle | 1.0 transmission clock cycle |
| $0010_B$ | 1.0 transmission clock cycle | 1.5 transmission clock cycles |
| $0011_B$ | 2.0 transmission clock cycles | 2.5 transmission clock cycles |
| $0100_B$ | 3.0 transmission clock cycles | 3.5 transmission clock cycles |
| $0101_B$ | 4.0 transmission clock cycles | 4.5 transmission clock cycles |
| $0110_B$ | 6.0 transmission clock cycles | 6.5 transmission clock cycles |
| $0111_B$ | 8.0 transmission clock cycles | 8.5 transmission clock cycles |
| $1000_B$ | 9.0 transmission clock cycles | 9.5 transmission clock cycles |
| $1001_B$ | 10.0 transmission clock cycles | 10.5 transmission clock cycles |
| $1010_B$ | 11.0 transmission clock cycles | 11.5 transmission clock cycles |
| $1011_B$ | 12.0 transmission clock cycles | 12.5 transmission clock cycles |
| $1100_B$ | 14.0 transmission clock cycles | 14.5 transmission clock cycles |
| $1101_B$ | 16.0 transmission clock cycles | 16.5 transmission clock cycles |
| $1110_B$ | 18.0 transmission clock cycles | 18.5 transmission clock cycles |
| $1111_B$ | 20.0 transmission clock cycles | 20.5 transmission clock cycles |

These bits are only available in master mode.

| | | |
|---|---|---|
| 3 to 0 | CSIHnSPx[3:0] | Specifies the setup time for chip select signal x in transmission clock cycles. |

| CSIHnSPx[3:0] | Setup Time |
|---|---|
| $0000_B$ | 0.5 transmission clock cycle |
| $0001_B$ | 1.0 transmission clock cycle |
| $0010_B$ | 1.5 transmission clock cycles |
| $0011_B$ | 2.5 transmission clock cycles |
| $0100_B$ | 3.5 transmission clock cycles |
| $0101_B$ | 4.5 transmission clock cycles |
| $0110_B$ | 6.5 transmission clock cycles |
| $0111_B$ | 8.5 transmission clock cycles |
| $1000_B$ | 9.5 transmission clock cycles |
| $1001_B$ | 10.5 transmission clock cycles |
| $1010_B$ | 11.5 transmission clock cycles |
| $1011_B$ | 12.5 transmission clock cycles |
| $1100_B$ | 14.5 transmission clock cycles |
| $1101_B$ | 16.5 transmission clock cycles |
| $1110_B$ | 18.5 transmission clock cycles |
| $1111_B$ | 20.5 transmission clock cycles |

These bits are only available in master mode.

**CAUTION**

When setting this register, see **Table 17.32, Notes on Setting Registers**.

## 17.3.12 CSIHnTX0W — CSIHn Transmit Data Register 0 for Word Access

This register stores transmission data. In addition, it specifies the communication interrupt request, the end-of-job, the extended data length, and the chip select activation.

**Access:** This register can be read or written in 32-bit units.

**Address:** <CSIHn_base> + $1008_H$

**Value after reset:** $X0XX\ XXXX_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnCIRE | CSIHnEOJ | CSIHnEDL | — | — | — | — | — | CSIHnCS7 | CSIHnCS6 | CSIHnCS5 | CSIHnCS4 | CSIHnCS3 | CSIHnCS2 | CSIHnCS1 | CSIHnCS0 |
| Value after reset | — | — | — | 0 | 0 | 0 | 0 | 0 | — | — | — | — | — | — | — | — |
| R/W | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnTX[15:0] | | | | | | | | | | | | | | | |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 17.26 CSIHnTX0W Register Contents (1/2)**

| Bit Position | Bit name | Function |
|---|---|---|
| 31 | CSIHnCIRE | Enables the communication interrupt request INTCSIHTIC in dual buffer or transmit-only buffer mode, or the job completion interrupt request INTCSIHTIJC in FIFO mode.<br>0: No interrupt is requested.<br>1: An interrupt is requested. Generates interrupt INTCSIHTIC or INTCSIHTIJC after transmission. For details, see **Section 17.4.3, INTCSIHTIC (Communication Status Interrupt)** and **Section 17.4.6, INTCSIHTIJC (Job Completion Interrupt)**.<br>**CAUTION**<br>This bit is only valid when job mode is enabled (CSIHnCTL1.CSIHnJE = 1). |
| 30 | CSIHnEOJ | Specifies the end of a job.<br>0: Indicates that it is not end-of-job data. The job continues.<br>1: Indicates end-of-job data.<br>**CAUTION**<br>This bit is only valid when job mode is enabled (CSIHnCTL1.CSIHnJE = 1). This bit must be set to 0 in slave mode. |
| 29 | CSIHnEDL | Specifies whether the associated data requires the extended data length (EDL) option.<br>0: Normal operation<br>1: Enables the extended data length.<br>The associated data is transmitted as a 16-bit packet. No inter-data time or idle time will be inserted after the data is transmitted.<br>If CSIHnCTL1.CSIHnEDLE = 1 and CSIHnTX0W.CSIHnEDL = 1, the subsequent data must have the same CS selection. If CS is modified for the subsequent data, the correct operation is not assured.<br>**CAUTION**<br>This bit is only available if CSIHnCTL1.CSIHnEDLE = 1. |
| 28 to 24 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

**Table 17.26 CSIHnTX0W Register Contents (2/2)**

| Bit Position | Bit name | Function |
|---|---|---|
| 23 to 16 | CSIHnCS[7:0] | Activates one or more chip select signals.<br>0: Activates chip select signals x for the associated transmission.<br>1: Deactivates chip select signals x for the associated transmission.<br>Setting CSIHnTX0W.CSIHnCS[7:0] = $FF_H$ is prohibited.<br>**CAUTION**<br>If several chip select signals are enabled for broadcasting, the configuration of one with CSIHnCFGx.CSIHnRCBx = 0 (dominant) is used. In this case, all dominant chip select signals must be set to precisely the same value.<br>In slave mode, set the CSIHnCS[7:0] bit to $FE_H$. |
| 15 to 0 | CSIHnTX[15:0] | Stores the transmission data. |

**CAUTION**

When setting this register, see **Table 17.32, Notes on Setting Registers**.

### 17.3.13 CSIHnTX0H — CSIHn Transmit Data Register 0 for Half Word Access

This register stores the transmission data. This register is the same as bits 15 to 0 of register CSIHnTX0W.

The settings specified by the upper 16 bits of CSIHnTX0W are applied to the transmission. Set transmit data to CSIHnTX0W before using this register because the value of CSIHnTX0W is undefined after the reset.

**Access:** This register can be read or written in 16-bit units.

**Address:** <CSIHn_base> + $100C_H$

**Value after reset:** Undefined

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnTX[15:0] | | | | | | | | | | | | | | | |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 17.27 CSIHnTX0H Register Contents**

| Bit Position | Bit name | Function |
|---|---|---|
| 15 to 0 | CSIHnTX[15:0] | Stores the transmission data. |

**CAUTION**

When setting this register, see **Table 17.32, Notes on Setting Registers**.

## 17.3.14 CSIHnRX0W — CSIHn Receive Data Register 0 for Word Access

This register stores the received data.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <CSIHn_base> + $1010_H$

**Value after reset:** $0XXX\ XXXX_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CSIHn RPE | CSIHn TDCE | CSIHn CS7 | CSIHn CS6 | CSIHn CS5 | CSIHn CS4 | CSIHn CS3 | CSIHn CS2 | CSIHn CS1 | CSIHn CS0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnRX[15:0] | | | | | | | | | | | | | | | |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 17.28 CSIHnRX0W Register Contents**

| Bit Position | Bit name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset is returned. |
| 25 | CSIHnRPE | Indicates whether a reception data parity error was detected.<br>0: No parity error was detected on the associated reception data.<br>1: A parity error was detected on the associated reception data. |
| 24 | CSIHnTDCE | Indicates whether a transmission data consistency check error was detected.<br>0: No consistency error was detected on the associated transmission data.<br>1: A consistency error was detected on the associated transmission data. |
| 23 to 16 | CSIHnCSx (x = 7 to 0) | Indicates which chip select signal was activated.<br>0: Chip select x was activated for the associated reception.<br>1: Chip select x was deactivated for the associated reception. |
| 15 to 0 | CSIHnRX[15:0] | Stores the received data. |

**NOTE**

Read this register at least one serial clock cycle before an interrupt is generated.

**CAUTION**

When setting this register, see **Table 17.32, Notes on Setting Registers**.

## 17.3.15 CSIHnRX0H — CSIHn Receive Data Register 0 for Half Word Access

This register stores the received data. This register is the same as bits 15 to 0 of register CSIHnRX0W.

**Access:** This register is a read-only register that can be read in 16-bit units.

**Address:** <CSIHn_base> + $1014_H$

**Value after reset:** Undefined

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnRX[15:0] | | | | | | | | | | | | | | | |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 17.29 CSIHnRX0H Register Contents**

| Bit Position | Bit name | Function |
|---|---|---|
| 15 to 0 | CSIHnRX[15:0] | Stores the received data. |

**CAUTION**

When setting this register, see **Table 17.32, Notes on Setting Registers**.

## 17.3.16 CSIHnEMU — CSIHn Emulation Register

This register controls operation of SVSTOP.

**Access:** This register can be read or written in 8-bit or 1-bit units.
Perform write operation when (EPC.SVSTOP = 0).

**Address:** <CSIHn_base> + $0018_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CSIHnSVSDIS | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R |

**Table 17.30 CSIHnEMU Register Contents**

| Bit Position | Bit name | Function |
|---|---|---|
| 7 | CSIHn SVSDIS | Selects whether to continue or stop transmit/receive operation during debugging.<br>• When the EPC.SVSTOP bit is set to 0<br>Continues transmit/receive operation regardless of the setting of this bit.<br>• When the EPC.SVSTOP bit is set to 1<br>0: Stops transmit/receive operation.<br>1: Continues transmit/receive operation. |
| 6 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

**CAUTION**

When setting this register, see **Table 17.32, Notes on Setting Registers**.

## 17.3.17 CSIHnBRSy — CSIHn Baud Rate Setting Register y (y = 0 to 3)

This register sets the transfer clock frequency for each chip select signal.

With CSIHnCFG0 to 7.CSIHnBRSSx[1:0] bits, one of the four types of transfer clock frequency settings can be selected for each chip select signal. For details of transfer clock frequency setting, see **Section 17.5.5, Transmission Clock Selection**.

**Access:** This register can be read or written in 16-bit units.

**Address:** CSIHnBRS0: <CSIHn_base> + $1068_H$
CSIHnBRS1: <CSIHn_base> + $106C_H$
CSIHnBRS2: <CSIHn_base> + $1070_H$
CSIHnBRS3: <CSIHn_base> + $1074_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | CSIHnBRS[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 17.31 CSIHnBRSy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 12 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 11 to 0 | CSIHnBRS [11:0] | 0: BRG stopped<br>1: PCLK / ($2^{\alpha}$ × 1 × 2)<br>2: PCLK / ($2^{\alpha}$ × 2 × 2)<br>3: PCLK / ($2^{\alpha}$ × 3 × 2)<br>4: PCLK / ($2^{\alpha}$ × 4 × 2)<br>.<br>.<br>.<br>4095: PCLK / ($2^{\alpha}$ × 4095 × 2)<br><br>α is the value of CSIHnCTL2.CSIHnPRS[2:0]. |

**CAUTION**

When setting this register, see **Table 17.32, Notes on Setting Registers**.

## 17.3.18 List of Cautions

Table 17.32 Notes on Setting Registers (1/3)

| Register Name | Bit Name | Cautions |
|---|---|---|
| CSIHnCTL0 | CSIHnPWR | If this bit is cleared during communication, ongoing communication is aborted. After the communication is aborted, it is necessary to restart the communication. |
| CSIHnCTL0 | CSIHnTXE<br>CSIHnRXE | Do not modify any of these bits while CSIHnCTL0.CSIHnPWR = 0. (These bits can be modified at the same time as the CSIHnCTL0.CSIHnPWR bit.)<br>Do not modify these bits while CSIHnSTR0.CSIHnTSF = 1, because the specified operation is not guaranteed if ongoing communication is aborted. |
| CSIHnCTL0 | CSIHnJOBE | Do not modify this bit while CSIHnCTL0.CSIHnPWR = 0.<br>This bit is only valid when CSIHnCTL1.CSIHnJE = 1.<br>Setting this bit is prohibited in slave mode. |
| CSIHnCTL0 | CSIHnMBS | Do not modify this bit while CSIHnCTL0.CSIHnPWR = 0. (This bit can be modified at the same time as the CSIHnCTL0.CSIHnPWR bit.)<br>Modification of this bit is only permitted while CSIHnSTR0.CSIHnTSF = 0.<br>Do not change the mode between FIFO mode and direct access mode while CSIHnCTL0.CSIHnPWR = 1.<br>When the CPU-controlled high-priority communication is enabled, the operation is the same as that in direct access mode regardless of the CSIHnMBS bit setting. |
| CSIHnCTL1 | CSIHnCKR | Modification of this bit is only permitted while CSIHnCTL0.CSIHnPWR = 0.<br>When CS is not used, use this bit instead of CSIHnCFGx.CSIHnCKPx and set CSIHnCFGx.CSIHnCKPx to 0.<br>This bit must be used in slave mode. |
| CSIHnCTL1 | CSIHnSLIT<br>CSIHnCSL[7:0]<br>CSIHnEDLE<br>CSIHnDCS<br>CSIHnCSRI<br>CSIHnHSE | Modification of these bits is only permitted while CSIHnCTL0.CSIHnPWR = 0. |
| CSIHnCTL1 | CSIHnPHE<br>CSIHnJE<br>CSIHnLBM | Modification of these bits is only permitted while CSIHnCTL0.CSIHnPWR = 0.<br>Setting of this bit is prohibited in slave mode. |
| CSIHnCTL1 | CSIHnSSE | Modification this bit is only permitted while CSIHnCTL0.CSIHnPWR = 0.<br>Setting this bit to 1 is prohibited in master mode. |
| CSIHnCTL1 | CSIHnSIT | Modification of this bit is only permitted while CSIHnCTL0.CSIHnPWR = 0.<br>This bit is only valid in master mode. In slave mode, no delay is generated. |
| CSIHnCTL2 | CSIHnPRS[2:0] | Modification of this bit is only permitted while CSIHnCTL0.CSIHnPWR = 0.<br>Setting of the maximum transfer clock frequency is as follows.<br>• Master mode: 10.0 MHz (however, it must be equal to or lower than PCLK/4)<br>• Slave mode: 5.0 MHz (however, it must be equal to or lower than PCLK/16) |
| CSIHnSTR0 | CSIHnSRP[7:0]<br>CSIHnSPF[7:0]<br>CSIHnHPST<br>CSIHnFLF<br>CSIHnEMF<br>CSIHnTSF | Writing to these bits is prohibited, and only reading is permitted. |
| CSIHnSTR0 | CSIHnTMOE<br>CSIHnOFE<br>CSIHnDCE<br>CSIHnPE<br>CSIHnOVE | Writing to these bits is prohibited, and only reading is permitted.<br>These bits are initialized when CSIHnCTL0.CSIHnPWR = 0 → 1 or CSIHnCTL0.CSIHnPWR = 1 → 0. |
| CSIHnSTCR0 | CSIHnPCT | If this bit is set to 1 during communication, ongoing communication is aborted. |
| CSIHnMCTL0 | CSIHnMMS[1:0] | Modification of these bits is only permitted while CSIHnCTL0.CSIHnPWR = 0 and CSIHnCTL0.CSIHnMBS = 0. |

**Table 17.32 Notes on Setting Registers (2/3)**

| Register Name | Bit Name | Cautions |
|---|---|---|
| CSIHnMCTL0 | CSIHnTO[4:0] | Modification of these bits is only permitted while CSIHnCTL0.CSIHnPWR = 0.<br>Set these bits to 0 in master mode.<br>Set these bits to 0 in direct access, dual buffer, and transmit-only buffer mode. |
| CSIHnMCTL1 | CSIHnFES[6:0]<br>CSIHnFFS[6:0] | Writing to these bits while communication is ongoing is permitted. |
| CSIHnMCTL2 | CSIHnBTST<br>CSIHnND[7:0]<br>CSIHnSOP[6:0] | Writing to these bits is prohibited when CSIHnCTL0.CSIHnPWR = 0.<br>Writing to these bits is prohibited when CSIHnCTL0.CSIHnTXE = CSIHnCTL0.CSIHnRXE = 0.<br>Writing to these bits is prohibited when CSIHnSTR0.CSIHnTSF = 1.<br>Writing to these bits is prohibited in direct access or FIFO mode. |
| CSIHnMRWP0 | CSIHnRRA[6:0] | Writing to these bits while communication is ongoing is permitted.<br>Writing to these bits is prohibited in direct access or FIFO mode.<br>When writing is required, set "$00_H$" to these bits in transmit-only buffer mode. |
| CSIHnMRWP0 | CSIHnTRWA[6:0] | Writing to these bits while communication is ongoing is permitted.<br>Writing to these bits is prohibited in direct access or FIFO mode. |
| CSIHnCFGx<br>x = 0 to 7 | CSIHnBRSSx[1:0]<br>CSIHnRCBx<br>CSIHnIDLx<br>CSIHnIDx[2:0]<br>CSIHnHDx[3:0]<br>CSIHnINx[3:0]<br>CSIHnSPx[3:0] | Modification of these bits is only permitted while CSIHnCTL0.CSIHnPWR = 0.<br>These bits must be set to 0 in slave mode. |
| CSIHnCFGx<br>x = 0 to 7 | CSIHnPSx[1:0]<br>CSIHnDLSx[3:0]<br>CSIHnDIRx<br>CSIHnDAPx | Modification of these bits is only permitted while CSIHnCTL0.CSIHnPWR = 0.<br>In slave mode, the CSIHnCFG0 setting is used for the configuration. Therefore, all the bits in CSIHnCFG1 to CSIHnCFG7 must be set to 0. |
| CSIHnCFGx<br>x = 0 to 7 | CSIHnCKPx | Modification of these bits is only permitted while CSIHnCTL0.CSIHnPWR = 0.<br>Set this bit to 0 as CSIHnCTL1.CSIHnCKR must be used in slave mode.<br>If CS is not used, the CSIHnCTL1.CSIHnCKR bit instead of this bit, and clear this bit to 0. |
| CSIHnTX0W | CSIHnEOJ<br>CSIHnCIRE | These bits are only valid when CSIHnCTL1.CSIHnJE = 1.<br>While CSIHnCTL1.CSIHnJE = 0, the values of these bits are ignored even if 1 is read.<br>Set these bits to 0 in slave mode. |
| CSIHnTX0W | CSIHnEDL | This bit is only valid when CSIHnCTL1.CSIHnEDLE = 1.<br>While CSIHnCTL1.CSIHnEDLE = 0, the value of this bit is ignored even if 1 is read. |
| CSIHnTX0W | CSIHnCS[7:0] | In master mode, setting this bit to "$FF_H$" is prohibited.<br>In slave mode, set this bit to "$FE_H$". |
| CSIHnTX0W<br>CSIHnTX0H | | Reading these bits while communication is ongoing is prohibited in FIFO mode.<br>While CSIHnCTL0.CSIHnPWR = 0, reading and writing to these bits is prohibited in FIFO mode.<br>While CSIHnCTL0.CSIHnTXE = CSIHnCTL0.CSIHnRXE = 0, writing to these bits are prohibited in direct access mode. |
| CSIHnRX0W | | These bits are initialized when CSIHnCTL0.CSIHnPWR = 0 → 1 or CSIHnCTL0.CSIHnPWR = 1→ 0.<br>While CSIHnCTL0.CSIHnPWR = 0, reading and writing these bits is prohibited in FIFO mode.<br>While CSIHnCTL0.CSIHnPWR = 0, reading and writing these bits is permitted in the mode (Transmit-only buffer, Dual buffer and Direct access modes) exited FIFO mode.<br>While CSIHnCTL0.CSIHnPWR = 1, writing these bits is prohibited, and reading these bits is permitted.<br>Writing is prohibited and only reading is permitted of these bits while CSIHnCTL0.CSIHnPWR = 1. |

**Table 17.32 Notes on Setting Registers (3/3)**

| Register Name | Bit Name | Cautions |
|---|---|---|
| CSIHnRX0H | | These bits are initialized when CSIHnCTL0.CSIHnPWR = 0 → 1 or CSIHnCTL0.CSIHnPWR = 1→ 0.<br>While CSIHnCTL0.CSIHnPWR = 0, reading and writing these bits is prohibited in FIFO mode.<br>Writing is prohibited and only reading is permitted of these bits while CSIHnCTL0.CSIHnPWR = 1.<br>While CSIHnCTL0.CSIHnPWR = 1, writing these bits is prohibited in the FIFO mode and reading these bits is prohibited in the FIFO mode.<br>In spite of CSIHnCTL0.CSIHnPWR value, writing is prohibited and only reading is permitted of these bits in the mode (Transmit-only buffer, Dual buffer and Direct access modes) exited FIFO mode. |
| CSIHnEMU | CSIHnSVSDIS | Modification of this bit is only permitted while SVSTOP = 0. |
| CSIHnBRSy<br>y = 0 to 3 | | Modification of these bits is only permitted while CSIHnCTL0.CSIHnPWR = 0. |

## 17.4 Interrupt Sources

CSIH can generate the following interrupt requests:

- INTCSIHTIC (communication status interrupt)
- INTCSIHTIR (reception status interrupt)
- INTCSIHTIRE (communication error interrupt)
- INTCSIHTIJC (job completion interrupt)

### 17.4.1 Overview

The communication error interrupt INTCSIHTIRE is generated when an error is detected. The generation of the other interrupts depends on the memory mode, the job mode, and – in case of the job completion interrupt INTCSIHTIJC –also the operating mode.

The job completion interrupt INTCSIHTIJC is only generated when job mode is enabled (CSIHnCTL1.CSIHnJE = 1). It is not available in slave mode.

The following table gives an overview.

**Table 17.33 Interrupt Generation**

| Memory Mode | Interrupt | Interrupt Source | |
|---|---|---|---|
| | | **Job Mode Disabled CSIHnCTL1.CSIHnJE = 0** | **Job Mode Enabled CSIHnCTL1.CSIHnJE = 1** |
| FIFO | INTCSIHTIC | Tx data empty*1 | Tx data empty*1 except job abort*4 |
| | INTCSIHTIR | Rx data full*2 *and* CSIHnCTL0.CSIHnRXE = 1 | Rx data full*2 *and* CSIHnCTL0.CSIHnRXE = 1 |
| | INTCSIHTIRE | Error detected | Error detected |
| | INTCSIHTIJC*3 | Not applicable | CSIHnTX0W.CSIHnCIRE = 1 (except Tx data empty), or job abort*4 |
| Transmit-only buffer | INTCSIHTIC | End of communication | CSIHnTX0W.CSIHnCIRE = 1 *and* (CSIHnCTL0.CSIHnJOBE = 0 *or* CSIHnTX0W.CSIHnEOJ = 0) |
| | INTCSIHTIR | Data received *and* CSIHnCTL0.CSIHnRXE = 1 | Data received *and* CSIHnCTL0.CSIHnRXE = 1 |
| | INTCSIHTIRE | Error detected | Error detected |
| | INTCSIHTIJC*3 | Not applicable | Job abort*4 |
| Dual buffer | INTCSIHTIC | End of communication | CSIHnTX0W.CSIHnCIRE = 1 *and* (CSIHnCTL0.CSIHnJOBE = 0 *or* CSIHnTX0W.CSIHnEOJ = 0) |
| | INTCSIHTIR | End of communication *and* CSIHnCTL0.CSIHnRXE = 1 | Data received *and* CSIHnCTL0.CSIHnRXE = 1 |
| | INTCSIHTIRE | Error detected | Error detected |
| | INTCSIHTIJC*3 | Not applicable | Job abort*4 |
| Direct access | INTCSIHTIC | One data transfer | One data transfer except the state of job abort*4 |
| | INTCSIHTIR | Data received *and* CSIHnCTL0.CSIHnRXE = 1 | Data received *and* CSIHnCTL0.CSIHnRXE = 1 |
| | INTCSIHTIRE | Error detected | Error detected |
| | INTCSIHTIJC*3 | Not applicable | Job abort*4 |

Note 1. “Tx data empty” refers to the FIFO fill level, defined by CSIHnMCTL1.CSIHnFES[6:0].

Note 2. “Rx data full” refers to the FIFO fill level, defined by CSIHnMCTL1.CSIHnFFS[6:0].
Note 3. INTCSIHTIJC is not available in slave mode.
Note 4. Job abort condition: CSIHnTX0W.CSIHnEOJ = 1 and CSIHnCTL0.CSIHnJOBE = 1
During high priority communication in transmit-only buffer mode, the operation is the same as that in direct access mode.

### 17.4.2 Interrupt Delay

In master mode, all interrupts generated by the master can be delayed by half a cycle of the transmission clock, CSIHTSCK. This is not possible in slave mode.

The delay is specified by setting CSIHnCTL1.CSIHnSIT = 1. (The setting of the CSIHnSIT bit is invalid in slave mode.)

The following example illustrates the interrupt delay function, assuming a setting of CSIHnCTL1.CSIHnSIT = 1 (interrupt delay enabled), CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0 (clock phase and data phase), and CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$ (data length 8 bits).

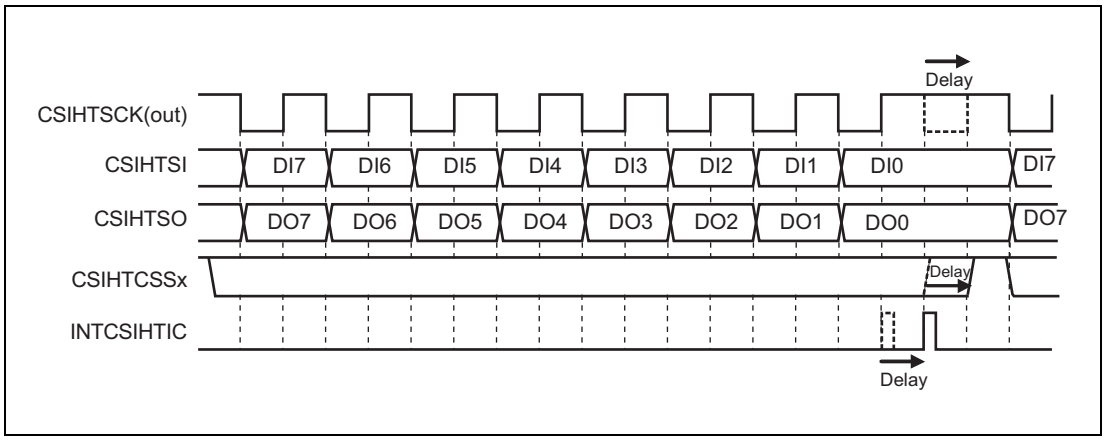



**Figure 17.2 Interrupt Delay Function (CSIHnCTL1.CSIHnSIT = 1)**

Setting CSIHnCTL1.CSIHnSIT = 1 adds a half cycle delay to the transmission clock. This also delays the end of the current chip select signal (CSIHTCSSx).

### 17.4.3 INTCSIHTIC (Communication Status Interrupt)

Depending on the memory mode and the job mode, this interrupt is generated according to the conditions shown in the following table.

**Table 17.34 INTCSIHTIC Interrupt Generation**

| Memory Mode | Interrupt Source | |
|---|---|---|
| | **Job Mode Disabled CSIHnCTL1.CSIHnJE = 0** | **Job Mode Enabled CSIHnCTL1.CSIHnJE = 1** |
| FIFO | This interrupt is generated just before transmission data in the FIFO runs out, notifying the application that new data should be added. INTCSIHTIC is generated if the number of transmit data remaining in the FIFO (CSIHnSTR0.CSIHnSPF[7:0]) equals CSIHnMTCL1.CSIHnFES[6:0]. | Similar to “when JE is 0”, an interrupt is generated when the number of transmit data remaining in the FIFO (CSIHnSTR0.CSIHnSPF[7:0]) equals CSIHnMCTL1.CSIHnFES[6:0]. However, it is not generated if a job is aborted. |
| Transmit-only buffer, dual buffer | An interrupt is generated at the end of communication. (specified by the CSIHnMTLC2.CSIHnND[7:0] bit) | Generated when data is transmitted while CSIHnTX0W.CSIHnCIRE = 1. Note that if data and job abort*1 are transmitted while CSIHnTX0W.CSIHnCIRE = 1, the INTCSIHTIJC interrupt is generated instead of INTCSIHTIC. |
| Direct access | Generated after every data transfer. | Generated after every data transfer, except when the communication was aborted. |

Note 1. Job abort condition: CSIHnTX0W.CSIHnEOJ = 1 and CSIHnCTL0.CSIHnJOBE = 1.
During high priority communication in transmit-only buffer mode, the operation is the same as that in direct access mode.

### 17.4.3.1 INTCSIHTIC in Direct Access Mode

The examples below show the INTCSIHTIC behavior in direct access mode.

The examples assume:

- Master mode
- Direct access mode
- No interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Normal clock phase and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- Data length 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$)
- Normal INTCSIHTIC interrupt timing (CSIHnCTL1.CSIHnSLIT = 0)

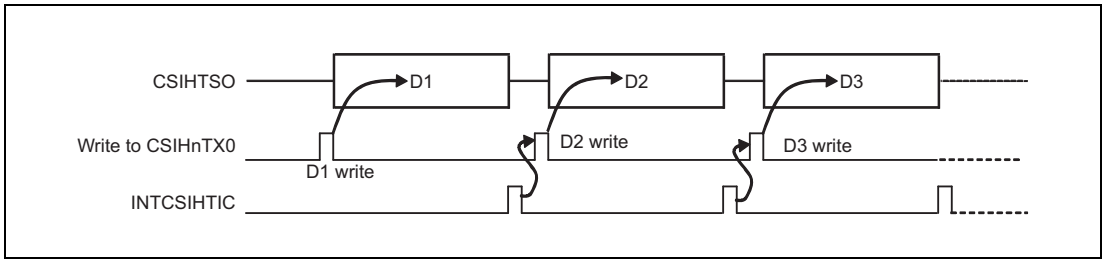



**Figure 17.3 Generation of INTCSIHTIC after Transfer (CSIHnCTL1.CSIHnSLIT = 0)**

If job mode is enabled (CSIHnCTL1.CSIHnJE = 1) and a job ends because data is sent with CSIHnTXOW.CSIHnEOJ = 1 and communication stop is requested (CSIHnCTL0.CSIHnJOBE = 1), then INTCSIHTIC is replaced by the job completion interrupt INTCSIHTIJC.

INTCSIHTIC can also be set up to occur as soon as the CSIHnTX0W/H register becomes empty and available for storing the next data. This is specified by setting CSIHnCTL1.CSIHnSLIT = 1.

The effect is illustrated in the figure below.

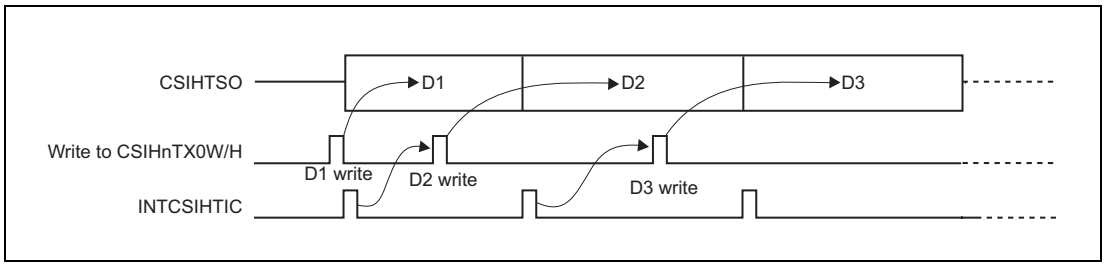



**Figure 17.4 Immediate Generation of INTCSIHTIC (CSIHnCTL1.CSIHnSLIT = 1)**

Thus, the new data can be written in advance.

**NOTE**

During high priority communication in transmit-only buffer mode, the operation is in the same as that in direct access mode.

### 17.4.3.2 INTCSIHTIC in FIFO Mode

The example below shows the INTCSIHTIC behavior in FIFO mode.

The example assumes:

- Master mode
- FIFO mode
- No interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Normal clock phase and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- Data length 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$)

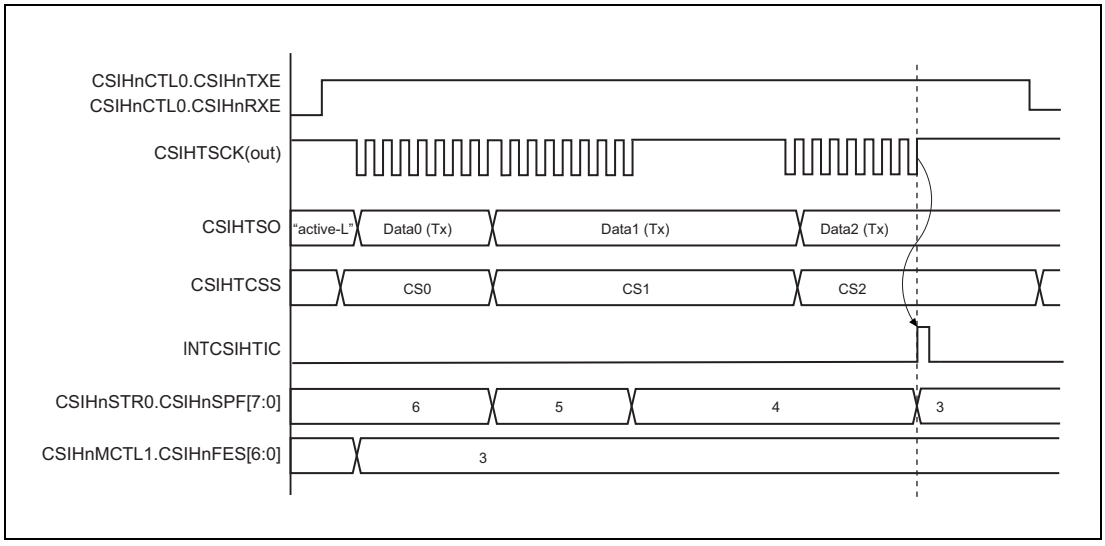



**Figure 17.5 Generation of INTCSIHTIC in FIFO Memory Mode**

The condition for "FIFO empty" is specified in CSIHnMCTL1.CSIHnFES[6:0]. In the example of the diagram above, the number of unsent data in FIFO is set to 3.

CSIHnSTR0.CSIHnSPF[7:0] indicates the number of unsent data. When both match, the interrupt INTCSIHTIC occurs.

### 17.4.3.3 INTCSIHTIC in Job Mode

The example below shows the INTCSIHTIC behavior in job mode.

The example assumes:

- Master mode
- Job mode enabled (CSIHnCTL1.CSIHnJE = 1)
- No interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Normal clock phase and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- Data length 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$)
- Normal INTCSIHTIC interrupt timing (CSIHnCTL1.CSIHnSLIT = 0)

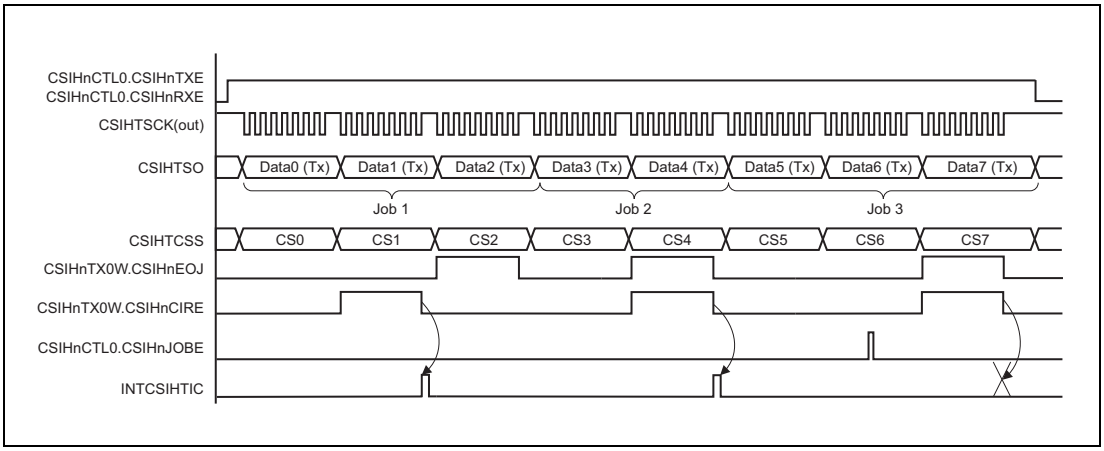



**Figure 17.6 Generation of INTCSIHTIC in Job Mode**

The rules for generating INTCSIHTIC in job mode are shown in the following table.

**Table 17.35 Generation of INTCSIHTIC in Job Mode**

| CSIHnTX0W. CSIHnEOJ | CSIHnTX0W. CSIHnCIRE | INTCSIHTIC |
|---|---|---|
| 0 | 0 | Not generated |
| 0 | 1 | Generated |
| 1 | 0 | Not generated |
| 1 | 1 | CSIHnCTL0.CSIHnJOBE = 0: Generated |
| | | CSIHnCTL0.CSIHnJOBE = 1: Not generated, replaced by interrupt INTCSIHTIJC |

### 17.4.4 INTCSIHTIR (Reception Status Interrupt)

Depending on the memory mode and the job mode, this interrupt is generated according to the conditions below.

**Table 17.36 INTCSIHTIR Interrupt Generation**

| Memory Mode | Interrupt Source | |
|---|---|---|
| | **Job Mode Disabled CSIHnCTL1.CSIHnJE = 0** | **Job Mode Enabled CSIHnCTL1.CSIHnJE = 1** |
| FIFO | This interrupt occurs when CSIHnCTL0.CSIHnRXE is 1 and the FIFO buffer is almost full with received data, notifying the application that the FIFO must be emptied. INTCSIHTIR is generated, if the number of received data in the FIFO (CSIHnSTR0.CSIHnSRP[7:0]) equals (128 – CSIHnMCTL1.CSIHnFFS[6:0]). | |
| Dual buffer | An interrupt is generated when the communication has finished (as specified by the CSIHnMCTL2.CSIHnND[7:0] bits) and CSIHnCTL0.CSIHnRXE = 1. | An interrupt is generated after every data transfer. |
| Transmit-only buffer, Direct access | An interrupt is generated after every data transfer. | |

#### 17.4.4.1 INTCSIHTIR in Direct Access Mode

The example below shows the INTCSIHTIR behavior in direct access mode.

The example below assumes:

- Master mode
- Direct access mode
- No interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Normal clock phase and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- Data length 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$)

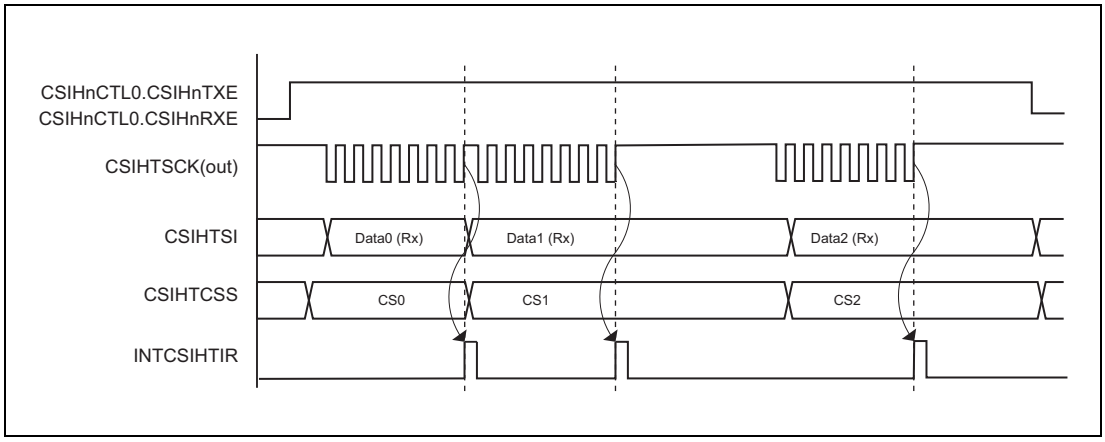



**Figure 17.7 Generation of INTCSIHTIR in Direct Access Mode**

### 17.4.4.2 INTCSIHTIR in Dual Buffer Mode

The example below shows the INTCSIHTIR behavior in dual buffer mode.

The example assumes:

- Master mode
- Dual buffer mode
- No interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Default clock phase and data phase
  (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- Data length 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$)

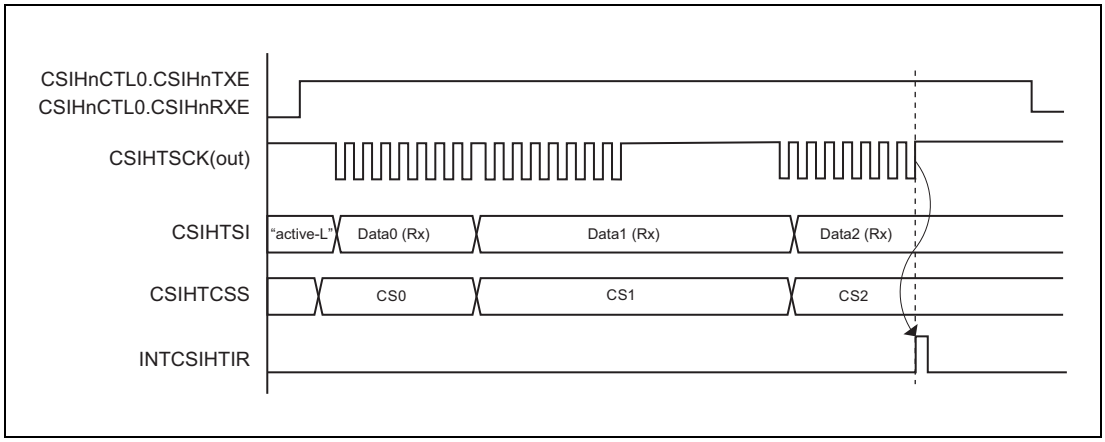



**Figure 17.8 Generation of INTCSIHTIR in Dual Buffer Mode**

### 17.4.5 INTCSIHTIRE (Communication Error Interrupt)

This interrupt is generated whenever an error is detected.

For details about interrupt generation timing, see **Section 17.5.12, Error Detection**.

**Table 17.37 Data Error Types**

| Error Type | Communication Status after Error Interrupt | Note |
|---|---|---|
| FIFO overflow error | Communication continues even if an interrupt is generated. | The data is not written to the FIFO buffer and the data that overflowed is lost, but communications started before the error continue. |
| Parity error | Communication continues even if an interrupt is generated. | — |
| Data consistency check error | Communication continues even if an interrupt is generated. | — |
| Time-out error | Communication continues even if an interrupt is generated. | — |
| Overrun error | **Condition for errors 1:**<br>In FIFO mode, when the number of received data is 0 and CPU reads the CSIHnRX0W/H register, an interrupt is generated and communication continues. | — |
| | **Condition for errors 2:**<br>In slave mode, when CSIHnCTL1.CSIHnHSE = 0 (handshake function disabled):<br>[1] In direct access mode or transmit-only buffer mode, when reception is completed while the previous received data is retained in the CSIHnRX0W/H register, an interrupt is generated, and communication continues.<br>[2] In FIFO mode, when reception by the FIFO buffer is completed and the buffer is in the full state, an interrupt is generated. Communication continues. | In slave mode, when CSIHnCTL1.CSIHnHSE = 1 (handshake function enabled), communication is suspended due to handshake, an overrun error is not generated. |

The type of error that caused the generation of INTCSIHTIRE is flagged in register CSIHnSTR0.

Additionally a parity error flag and a data consistency check error flag are attached to the received data in CSIHnRX0W.

For details about the various error types, see **Section 17.5.12, Error Detection**.

### 17.4.6 INTCSIHTIJC (Job Completion Interrupt)

This interrupt supports the handling of jobs. See **Section 17.5.3.3, Job Concept**. This interrupt is only available in master mode.

Job mode is enabled by setting CSIHnCTL1.CSIHnJE = 1. When CSIHnCTL1.CSIHnJE = 0, INTCSIHTIJC is not generated.

Depending on the memory mode, this interrupt is generated according to the condition shown in the following table.

**Table 17.38 INTCSIHTIJC Interrupt Generation**

| Memory Mode | Job Mode Disabled CSIHnCTL1.CSIHnJE = 0 | Job Mode Enabled CSIHnCTL1.CSIHnJE = 1 |
|---|---|---|
| FIFO | Not applicable | Indicates that the communication stopped at the end of a job after a job abort*1 was triggered<br>If FIFO empty is not detected, INTCSIHTIJC is generated when CSIHnCIRE is 1. |
| Transmit-only buffer | | Indicates that the communication stopped at the end of a job after a job abort*1 was triggered. |
| Dual buffer | | |
| Direct access | | |

Note 1. Job abort condition: CSIHnTX0W.CSIHnEOJ = 1 and CSIHnCTL0.CSIHnJOBE = 1

# 17.5 Operation

## 17.5.1 Operating Modes (Master/Slave)

Whether CSIH operates in the master or slave mode determines the source of the serial clock.

### 17.5.1.1 Master Mode

In master mode, the serial transmission clock is generated by the internal baud rate generator (BRG) and supplied to the slave(s) by signal CSIHTSCK.

Master mode is enabled by setting CSIHnCTL2.CSIHnPRS[2:0] to values other than $111_B$. In master mode, the BRG frequency can be specified by setting the CSIHnCTL2.CSIHnPRS[2:0] bits in combination with the CSIHnBRSy.CSIHnBRS[11:0] bits.

#### (1) Chip select signals

In master mode, one or more chip select signals can be used. If several slaves are connected to the master, the chip select signals can be used to select one or more slaves. Only the selected slave is then enabled for communication.

The communication protocol as well as additional parameters are stored separately for each chip select signal. This makes it possible to adapt the data transfer individually to the requirements of each slave. For details, see **Section 17.5.3, Chip Selection (CS) Features**.

#### (2) Clock defaults

The default level of CSIHTSCK depends on the clock phase inversion function bit of the CSIHTSCK, and is high when CSIHnCTL1.CSIHnCKR = 0 and is low when CSIHnCTL1.CSIHnCKR = 1.

The example below shows the communication in master mode for 8-bit data, CSIHnCTL1.CSIHnCKR = 0, CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0, and MSB first.

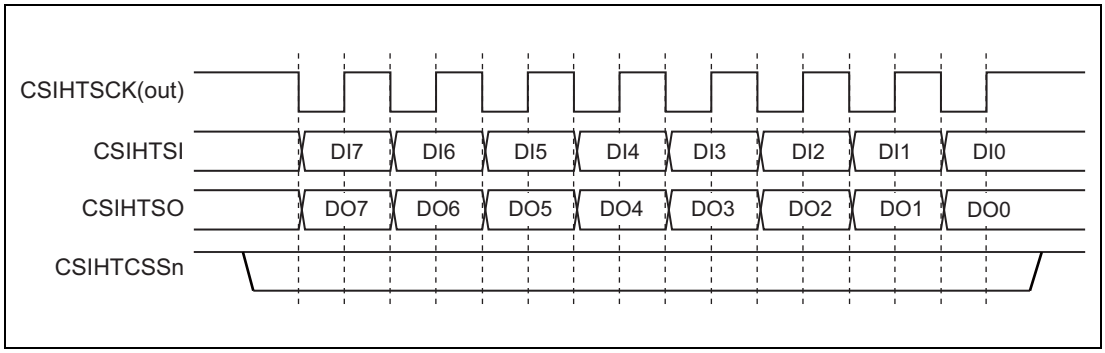



**Figure 17.9 Transmission/Reception in Master Mode**

### 17.5.1.2 Slave Mode

In slave mode, another device is the communication master and supplies the transmission clock. Normal transmit/receive operation is started as soon as a clock signal is detected.

Slave mode is selected by setting the CSIHnCTL2.CSIHnPRS[2:0] bits to $111_B$.

In slave mode, the transmission protocol setting of the CSIHnCFG0 register is enabled (settings of the CSIHnCFG1 - CSIHnCFG7 registers are disabled).

- CSIHnPSx[1:0]: parity usage
- CSIHnDLSx[3:0]: data length selection
- CSIHnDIRx: data direction
- CSIHnCKPx, CSIHnDAPx: clock phase and data phase

**NOTE**

When using slave mode, disable the baud rate generator (BRG) by setting the CSIHnBRSy.CSIHnBRS[11:0] bits to $000_H$. However, if you are using a time-out error, set the CSIHnBRSy.CSIHnBRS[11:0] bits to a value other than $000_H$.

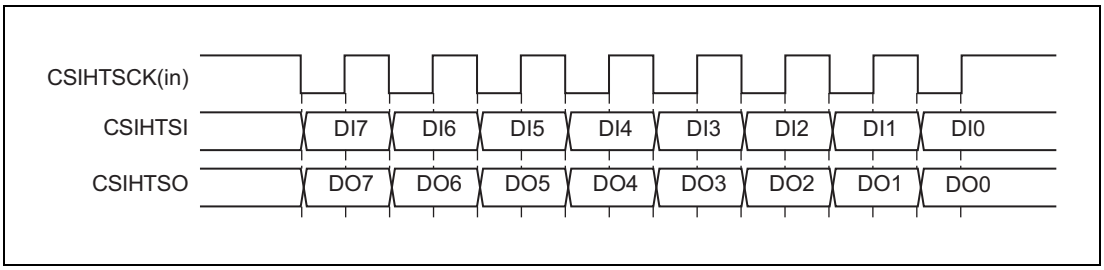



**Figure 17.10 Transmission/Reception in Slave Mode**

## 17.5.2 Master/Slave Connections

### 17.5.2.1 One Master and One Slave

The following figure illustrates the connections between one master and one slave.

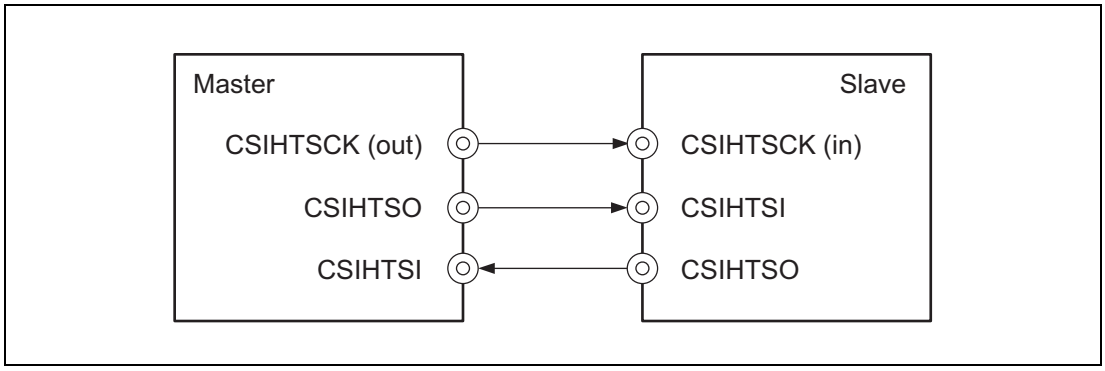



**Figure 17.11 Direct Master/Slave Connection**

### 17.5.2.2 One Master and Multiple Slaves

The following figure illustrates the connections between one master and multiple slaves. In this example, the master supplies one chip select (CS) signal to each of the slaves. This signal is connected to the slave select input $\overline{\text{CSIHTSSI}}$ of the slave.

The $\overline{\text{CSIHTSSI}}$ signal can be enabled/disabled by using the CSIHnCTL1.CSIHnSSE bit.

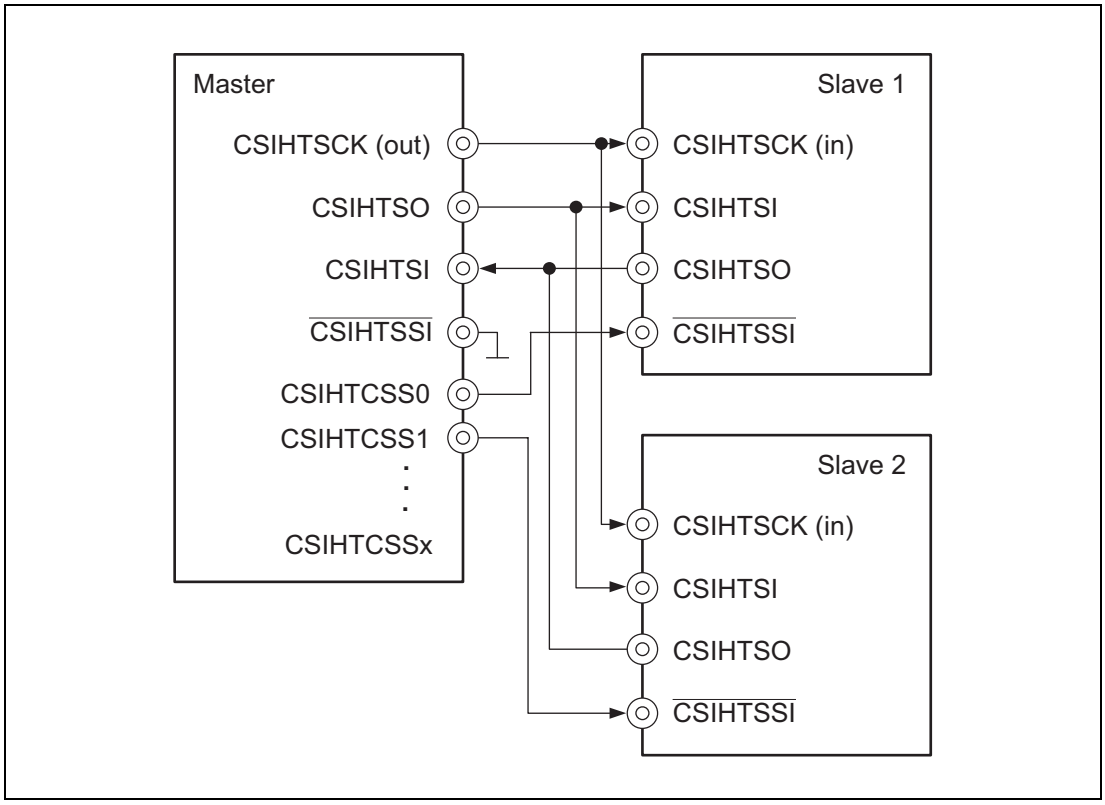



**Figure 17.12 Connections between One Master and Multiple Slaves**

By default, the chip select level is active low. That means, a slave is selected (enabled) as a CSIH slave when its $\overline{\text{CSIHTSSI}}$ signal is low level. However, to adapt the CS to other devices, the output level of each chip select signal can also be programmed to be active high.

If a slave is not selected, it will neither receive nor transmit data. In addition, when transmit-only mode or transmit/receive mode is set (CSIHnCTL0.CSIHnTXE = 1), the CSIHTSO output of the slaves that are not selected is disabled and set to input mode in order to avoid interference with the output of the selected slave.

### 17.5.3 Chip Selection (CS) Features

The chip select signal, CSIHTCSSx can be used by the master to select one or more slaves for communication.

#### 17.5.3.1 Configuration Registers

The parameters for each chip select signal CSIHTCSSx are defined in the corresponding configuration register CSIHnCFGx. The parameters include the communication protocol and additional CS parameters.

The communication protocol specifies:

- Data length: The number of bits to be sent or received.
  (CSIHnCFGx.CSIHnDLSx[3:0])
- Transfer direction: MSB or LSB first.
  (CSIHnCFGx.CSIHnDIRx)
- Parity usage: Odd, even, 0 parity or none.
  (CSIHnCFGx.CSIHnPSx[1:0])
- Clock phase and data phase.
  (CSIHnCFGx.CSIHnCKPx, CSIHnCFGx.CSIHnDAPx)

Additional parameters for each chip select signal that are only available in master mode are:

- Individual selection of the baud rate generator prescaler for each chip select signal
  (CSIHnCFGx.CSIHnBRSSx[1:0])
- Chip select priority: Categorizes chip select signals into “dominant” and “recessive”. The priority applies if two or more chip selects signals with different configurations are simultaneously activated for message broadcasting. In this case, the configuration specified for dominant chip select signals is used. (CSIHnCFGx.CSIHnRCBx)

The principle is also called “Recessive Configuration for Broadcasting” (RCB).

**CAUTION**

**When specifying multiple chip select signals as dominant, be sure to configure the same settings for all dominant signals.**

- Chip select timing:
  - Setup time $T_{setup}$: The time from when the CS signal becomes active to the start of data output.
    (CSIHnCFGx.CSIHnSPx[3:0])
  - Inter-data time $T_{inter}$: The time between one data and the next data while the same CS signal is active.
    (CSIHnCFGx.CSIHnINx[3:0])
  - Hold time $T_{hold}$: The time during which the CS signal remains active until CS is switched.
    (CSIHnCFGx.CSIHnHDx[3:0])
  - Idle time $T_{idle}$: Inactive time after terminating a CS signal or after every data transfer to the same CSx. (CSIHnCFGx.CSIHnIDx[2:0])

The CS timings of the setup time, the inter-data time, the hold time, and the idle time are illustrated in the figure below. When CSIHnCFGx.CSIHnIDLx bit is set to 1, IDLE time is inserted for every transfer regardless of CS signal.

**Figure 17.13** provides an example of when the default active low setting is specified for the CSIHTCSS1 and CSIHTCSS2 signals (CSIHnCTL1.CSIHnCSL1 bit = 0, CSIHnCTL1.CSIHnCSL2 bit = 0). The active level can be specified individually for each CS.

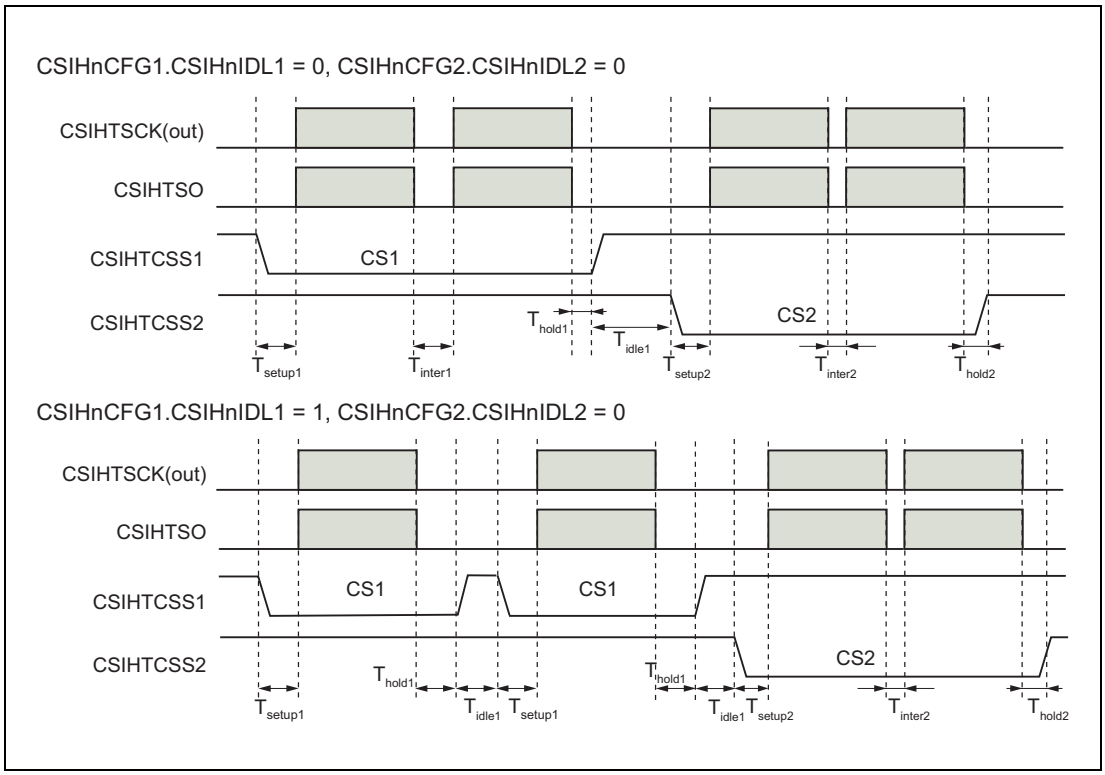



**Figure 17.13 Chip Select Timings**

Note that each CS signal can have a different value for the setup time, inter-data time, hold time, and idle time.

A particular chip select signal is activated by setting the appropriate bit in the transmission register CSIHnTX0W.CSIHnCSx.

CSIHnRX0W.CSIHnCSx in the reception register indicates the chip select signal associated with the received data.

**CAUTION**

**When high priority communication function by CPU control is enabled (CSIHnCTL1.CSIHnPHE = 1), IDLE state is inserted regardless of IDLn bit settings when priority communication mode is changed from low to high and from high to low.**

### 17.5.3.2 CS Example

The following figure shows an example of two consecutive data transmissions.

The first communication uses CS0 to communicate with a single slave. The second enables CS0 and CS1 to broadcast a message to two slaves. The priority of CS0 is set to “recessive: low priority” and the priority of CS1 to “dominant: high priority”. Consequently, the second communication is conducted using the CS1 settings, which are set as dominant.

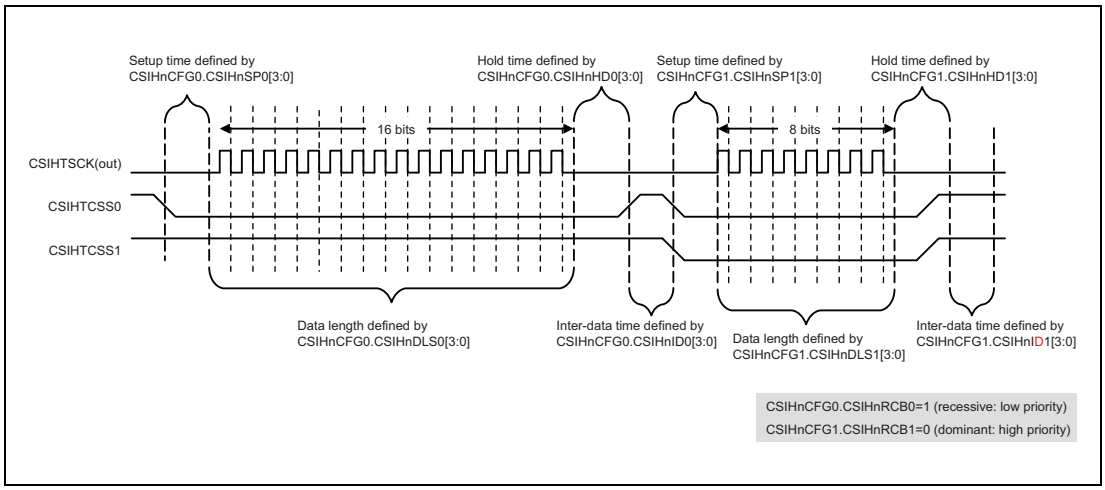



**Figure 17.14 Chip Select and RCB Example**

### 17.5.3.3 Job Concept

In CSIH, a job consists of the number of data targeted for transfer.

#### Job mode enable

The job mode can only be enabled in master mode. The job mode is enabled or disabled by CSIHnCTL1.CSIHnJE, while the CSIH is disabled by CSIHnCTL0.CSIHnPWR = 0.

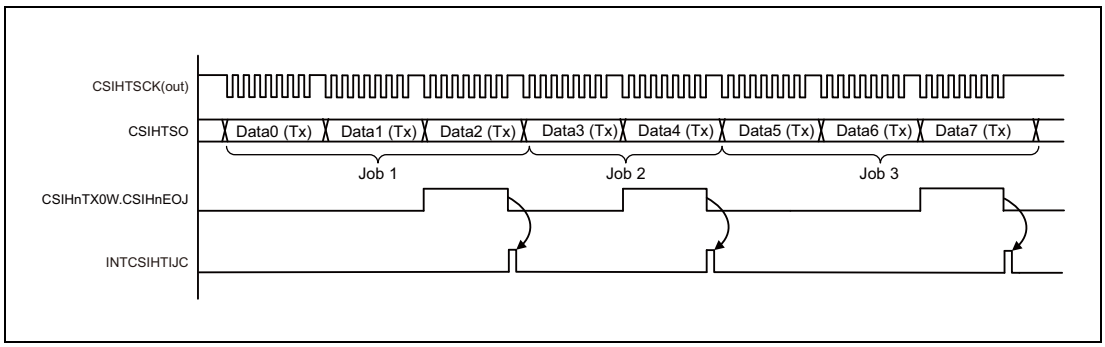



**Figure 17.15 Job Examples**

A job ends by transmitting data with CSIHnTX0W.CSIHnEOJ = 1.

Communication can be specified to stop when a job is finished. This is done by setting CSIHnCTL0.CSIHnJOBE. When CSIHnJOBE is set, the communication continues until the data for which the CSIHnEOJ bit is set is transmitted. After this data is sent, the communication is stopped and the job completion interrupt INTCSIHTIJC is generated.

## 17.5.4 Details of Chip Select Timing

### 17.5.4.1 Changing the Clock Phase

The serial clock level specified by CSIHnCFGx.CSIHnCKPx can be changed while communication is stopped. The minimum value of an idle time is one transmission clock (CSIHTSCK(out)) cycle.

If the idle time is set to 0.5 transmission clock cycles (in CSIHnCFGx.CSIHnIDx[2:0]) and two consecutive data is sent with different CSIHnCFGx.CSIHnCKPx configurations, the idle time is automatically extended to one CSIHTSCK(out) cycle.

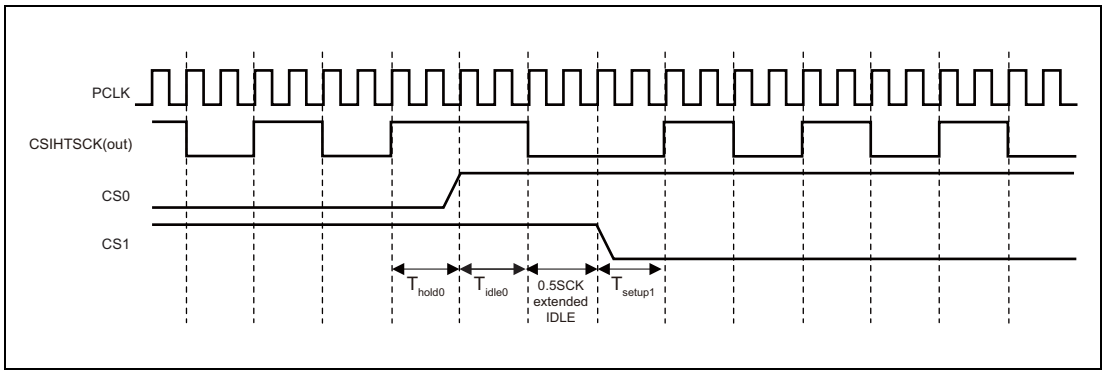



**Figure 17.16 Clock Phase Timing with PCLK/4, $T_{hold}0 = T_{setup1}$ = 0.5CSIHTSCK, $T_{idle0}$ = 0.5CSIHTSCK, CSIHnCFG0.CSIHnCKP0 = 0 (CSIHTCSS0) → CSIHnCFG1.CSIHnCKP1 = 1 (CSIHTCSS1)**

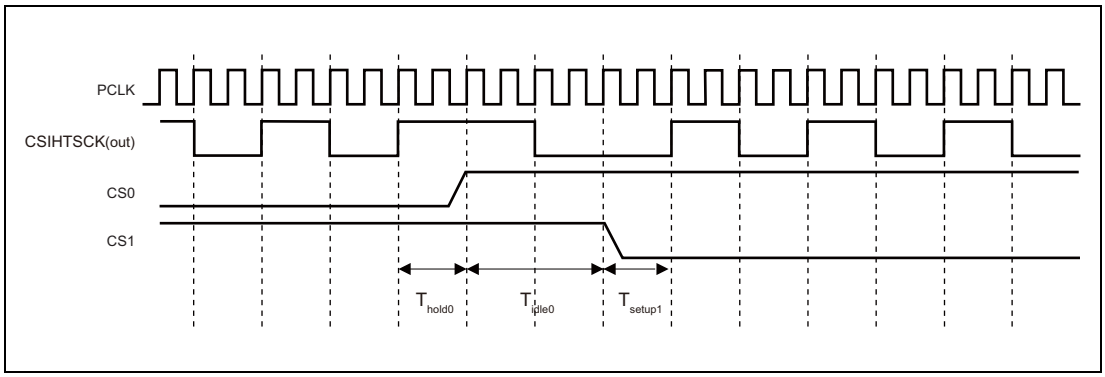



**Figure 17.17 Clock Phase Timing with PCLK/4, $T_{hold}0 = T_{setup1}$ = 0.5CSIHTSCK, $T_{idle0}$ = 1CSIHTSCK, CSIHnCFG0.CSIHnCKP0 = 0 (CSIHTCSS0) → CSIHnCFG1.CSIHnCKP1 = 1 (CSIHTCSS1)**

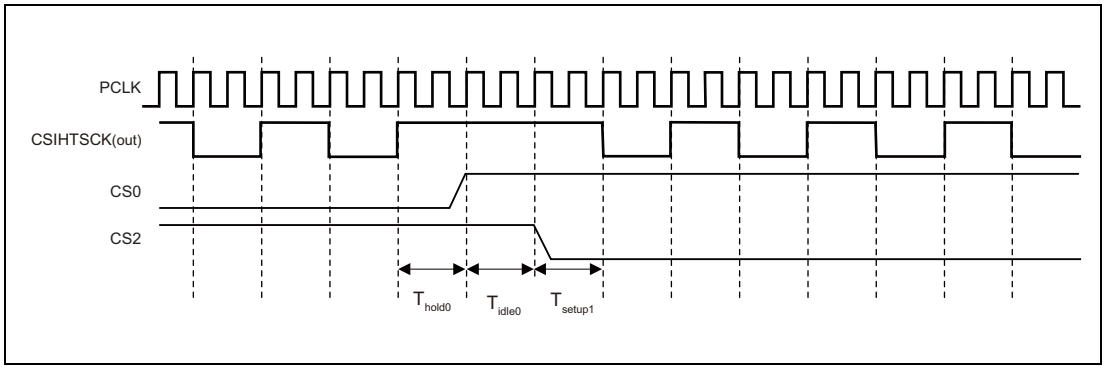



**Figure 17.18 Clock Phase Timing with PCLK/4, $T_{hold}0 = T_{setup1}$ = 0.5CSIHTSCK, $T_{idle0}$ = 0.5CSIHTSCK, CSIHnCFG0.CSIHnCKP0 = 0 (CSIHTCSS0) → CSIHnCFG2.CSIHnCKP2 = 0 (CSIHTCSS2)**

### 17.5.4.2 Changing the Data Phase

The CSIHnCFGx.CSIHnDAPx bit defines the phase of the data bits relative to the clock.

The relation between the setting of the CSIHnCFGx.CSIHnDAPx bit and the hold and setup times is as follows:

Hold time is the period from the last edge of the serial clock (CSIHTSCK) until the signals on CSIHTCSS[7:0] change to the inactive level.

Setup time is the period from when the signals on CSIHTCSS[7:0] change to the active level until to when the transmission data (CSIHTSO) is output.

Therefore, there is a gap of 0.5 CSIHTSCK cycles until the edge of the serial clock signal (CSIHTSCK) is output according to the CSIHnCFGx.CSIHnDAPx setting.

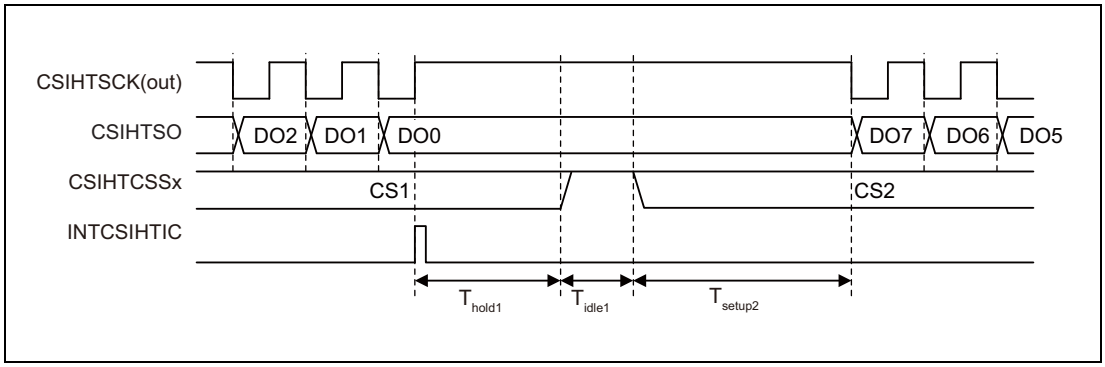



**Figure 17.19 Data Phase Timing with CSIHnCFG1.CSIHnCKP1 = 0, CSIHnCFG1.CSIHnDAP1 = 0 and CSIHnCFG2.CSIHnCKP2 = 0, CSIHnCFG2.CSIHnDAP2 = 0**

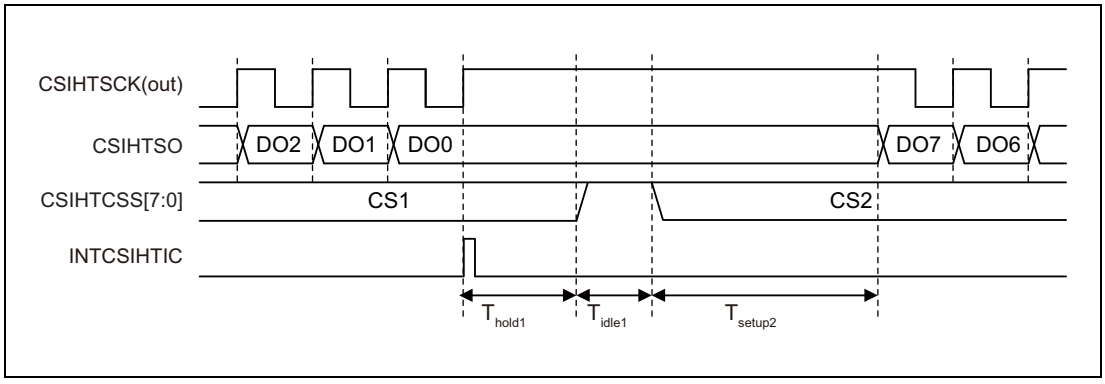



**Figure 17.20 Data Phase Timing with CSIHnCFG1.CSIHnCKP1 = 1, CSIHnCFG1.CSIHnDAP1 = 0 and CSIHnCFG2.CSIHnCKP2 = 0, CSIHnCFG2.CSIHnDAP2 = 1**

### 17.5.5 Transmission Clock Selection

In master mode, the transfer clock frequency is selectable using the following bits:

- CSIHnCTL2.CSIHnPRS[2:0]
- CSIHnBRSy.CSIHnBRS[11:0]
- CSIHnCFGx.CSIHnBRSSx[1:0]

The transfer clock frequency of transmission clock CSIHTSCK is determined by the setting of the CSIHnCTL2.CSIHnPRS[2:0] bits and the setting of the CSIHnBRSy.CSIHnBRS[11:0] bits, but any one of CSIHnBRS3 to CSIHnBRS0 can be selected for each chip select signal by using the CSIHnCFGx.CSIHnBRSSx[1:0] bits.

The following figure shows a block diagram of the baud rate generator.

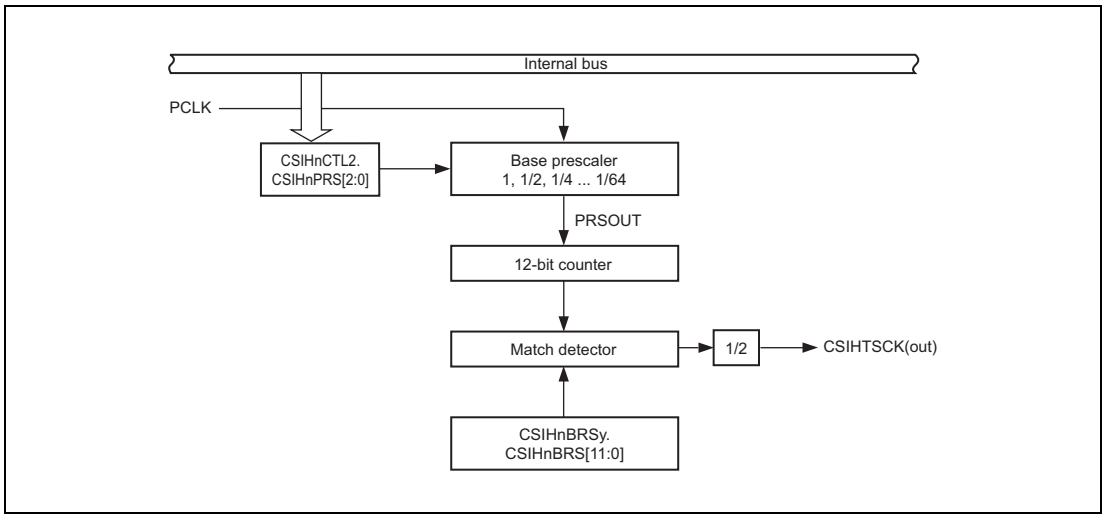



**Figure 17.21 Baud Rate Generator Block Diagram**

By setting CSIHnBRSy.CSIHnBRS[11:0] to $000_H$, the baud rate generator is disabled and all CSIHTSCK are stopped.

#### Transfer clock frequency calculation

The transfer clock frequency in master mode is calculated as:

Transfer clock frequency (CSIHTSCK) = PCLK / (division ratio of PCLK) = PCLK / ($2^{\alpha} \times k \times 2$),

where:

α = CSIHnCTL2.CSIHnPRS[2:0] = 0 to 6

k = CSIHnBRS0.CSIHnBRS0[11:0] = 1 to 4095
(when CSIHnCFGx.CSIHnBRSSx[1:0] = 0)
CSIHnBRS1.CSIHnBRS1[11:0] = 1 to 4095
(when CSIHnCFGx.CSIHnBRSSx[1:0] = 1)
CSIHnBRS2.CSIHnBRS2[11:0] = 1 to 4095
(when CSIHnCFGx.CSIHnBRSSx[1:0] = 2)
CSIHnBRS3.CSIHnBRS3[11:0] = 1 to 4095
(when CSIHnCFGx.CSIHnBRSSx[1:0] = 3)

### Transfer clock frequency upper and lower limits

When setting the transfer clock frequency, please note the following:

- The minimum transfer clock frequency in master mode and slave mode is PCLK/524160.
- The maximum transfer clock frequency is as follows:
  - In master mode: 10.0 MHz (however, it must be equal to or lower than PCLK/4)
  - In slave mode: 5.0 MHz (however, it must be equal to or lower than PCLK/16)

### 17.5.6 CSIH Buffer Memory

The CSIH has a configurable RAM that can be used for buffered I/O. The size is 128 words. One word is comprised of 32-bit data and 7-bit ECC.

The following configurations are available:

| Mode | CSIHnCTL0. CSIHnMBS | CSIHnMCTL0. CSIHnMMS[1:0] |
|---|---|---|
| FIFO mode | 0 | $00_B$ |
| Dual buffer mode | | $01_B$ |
| Transmit-only buffer mode | | $10_B$ |
| Direct access mode | 1 | X |

#### 17.5.6.1 FIFO Mode

In FIFO mode, data can be written to the CSIHnTX0W register without waiting for completion of the transmission, and data can be received without reading the CSIHnRX0W register immediately, provided the FIFO is not full.

Data to be transmitted is stored in the FIFO memory. Transmission and reception occur simultaneously – one data is received as one data is transmitted. That means, received data overwrites the transmitted data in the FIFO.

The CSIH automatically updates the respective FIFO memory pointers when data is written to or read from the FIFO memory, or data is transmitted to or received from the FIFO memory.

**Table 17.39 FIFO Mode**

| Pointer Description | Pointer*1 | Range |
|---|---|---|
| Number of untransmitted words | CSIHnSTR0.CSIHnSPF[7:0] | 0 to 128 |
| Number of words received and stored in the FIFO | CSIHnSTR0.CSIHnSRP[7:0] | 0 to 128 |
| Address for write/read of transmit data | CSIHnMRWP0.CSIHnTRWA[6:0] | $0000_H$ to $01FC_H$ |
| Address for read of received data | CSIHnMRWP0.CSIHnRRA[6:0] | $0000_H$ to $01FC_H$ |
| Address to be sent | CSIHnMCTL2.CSIHnSOP[6:0] | $0000_H$ to $01FC_H$ |

Note 1. The values are automatically updated after each read/write or data transmit/receive operation.

The CSIH status register contains two FIFO status flags:

- CSIHnSTR0.CSIHnFLF: FIFO full
- CSIHnSTR0.CSIHnEMF: FIFO empty

When this mode is started, bit CSIHnSTCR0.CSIHnPCT must be set. By doing this, CSIHnSTR0.CSIHnEMF is not reset, but set.

All FIFO pointers and FIFO flags except CSIHnSTR0.CSIHnEMF are reset and CSIHnSTR0.CSIHnEMF is set.

### 17.5.6.2 Dual Buffer Mode

In this mode, the memory is divided into two parts of equal size; 64 words are allocated to transmission data and 64 words to received data. In dual buffer mode, the respective buffer pointers indicate the values shown in the following table.

**Table 17.40 Dual Buffer Mode**

| Pointer Description | Pointer*1 | Range |
|---|---|---|
| Address of data written to and read from transmit buffer | CSIHnMRWP0.CSIHnTRWA[6:0] | $0000_H$ to $00FC_H$ |
| Address of data read from receive buffer | CSIHnMRWP0.CSIHnRRA[6:0] | $0000_H$ to $00FC_H$ |
| The number of transmit data remaining in the transmit buffer | CSIHnMCTL2.CSIHnND[6:0] | 0 to 64 |
| Address to which data is transmitted | CSIHnMCTL2.CSIHnSOP[6:0] | $0000_H$ to $00FC_H$ |

Note 1. Pointers are automatically incremented after each read/write.

### 17.5.6.3 Transmit-Only Buffer Mode

In this mode, the entire memory is used to save transmission data.

Received data must be read directly from CSIHnRX0W/H.

In transmit-only buffer mode, the respective buffer pointer indicates the values shown in the following table.

**Table 17.41 Transmit-Only Buffer Mode**

| Pointer Description | Pointer*1 | Range |
|---|---|---|
| Address of data written to and read from transmit buffer | CSIHnMRWP0.CSIHnTRWA[6:0] | $0000_H$ to $01FC_H$ |
| The number of transmit data remained in the transmit buffer | CSIHnMCTL2.CSIHnND[6:0] | 0 to 128 |
| Address to which data is transmitted | CSIHnMCTL2.CSIHnSOP[6:0] | $0000_H$ to $01FC_H$ |

Note 1. Pointers are automatically incremented after each read/write.

### 17.5.6.4 Direct Access Mode

In direct access mode, the CSIH memory is completely bypassed:

- Transmission data provided by the CPU to the transmission register CSIHnTX0W or CSIHnTX0H is directly copied to the shift register.
- Reception data is directly copied from the shift register to the reception register CSIHnRX0W or CSIHnRX0H.

## 17.5.7 Data Transfer Modes

### 17.5.7.1 Transmit-Only Mode

Setting CSIHnCTL0.CSIHnTXE = 1 and CSIHnCTL0.CSIHnRXE = 0 puts the CSIH in transmit-only mode. Start of transmission depends on the memory mode:

- In case of FIFO or direct access mode, transmission starts when transmit data is written to the CSIHnTX0W or CSIHnTX0H register.
- In case of dual buffer or transmit-only buffer mode, transmission starts when the CSIHnMCTL2.CSIHnBTST bit is set.

### 17.5.7.2 Receive-Only Mode

Setting CSIHnCTL0.CSIHnTXE = 0 and CSIHnCTL0.CSIHnRXE = 1 puts the CSIH in receive-only mode.

In master mode, start of reception depends on the memory mode:

- In case of FIFO or direct access mode, reception starts when dummy data is written to the CSIHnTX0W or CSIHnTX0H register.

In slave mode, reception starts as soon as the CSIHTSCK transmission clock is received from the master. It is not necessary to write data to the CSIHnTX0W or CSIHnTX0H register of the slave.

- In case of dual buffer mode or transmit-only buffer mode, reception starts when the CSIHnMCTL2.CSIHnBTST bit is set.

### 17.5.7.3 Transmit/Receive Mode

Setting CSIHnCTL0.CSIHnTXE = 1 and CSIHnCTL0.CSIHnRXE = 1 puts the CSIH in transmit/receive mode.

Start of communication (transmission and reception) depends on the memory mode:

- In case of FIFO or direct access mode, communication starts when transmit data is written to the CSIHnTX0W or CSIHnTX0H register.
- In case of dual buffer or transmit-only buffer mode, communication starts when the CSIHnMCTL2.CSIHnBTST bit is set.

### 17.5.7.4 Summary

The following table summarizes this section. It shows how data transfer is started in the various memory, operating, and transfer modes.

**Table 17.42 Start of Data Transfer**

| Memory and Operating Mode | | Transfer Mode | |
|---|---|---|---|
| | | Transmit-Only<br>Transmit/Receive | Receive-Only |
| FIFO, direct access | Master | Writing to the CSIHnTX0W register or the CSIHnTX0H register | Writing to the CSIHnTX0W register or the CSIHnTX0H register |
| | Slave | Clock reception from master | Incoming clock from the master |
| Transmit-only buffer, dual buffer | Master | CSIHnMCTL2.CSIHnBTST = 1 | CSIHnMCTL2.CSIHnBTST = 1 |
| | Slave | Clock reception from master | Clock reception from master |

## 17.5.8 Data Length Selection

### 17.5.8.1 Data Length from 2 to 16 Bits

The length of a data packet is selectable for each chip select signal from 2 to 16 bits using CSIHnCFGx.CSIHnDLSx[3:0]. The examples below show the communication with MSB first (CSIHnCFGx.CSIHnDIRx = 0).

Data length = 16 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $0000_B$):

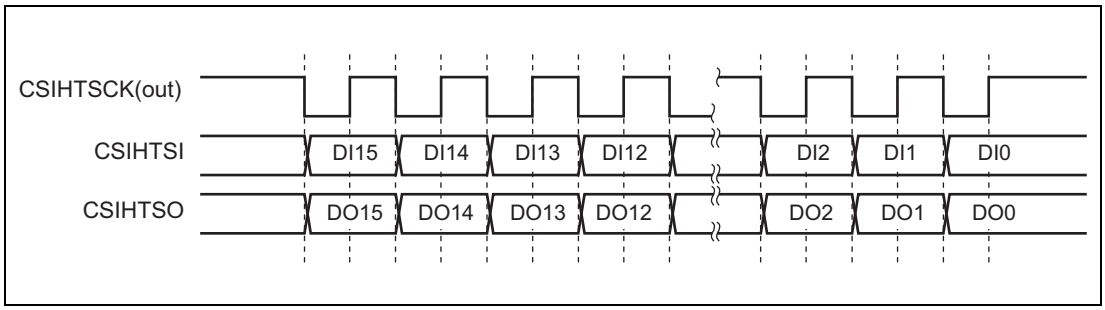



**Figure 17.22 16 Bit Data Length, MSB First**

Data length = 14 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1110_B$):

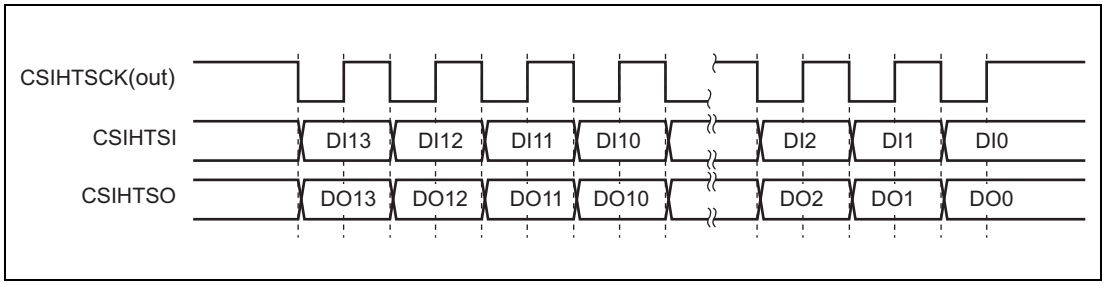



**Figure 17.23 14 Bit Data Length, MSB First**

### 17.5.8.2 Data Length Greater than 16 Bits

If the data to be sent/received exceeds 16 bits, the extended data length (EDL) function can be used.

EDL function is enabled by setting the CSIHnCTL1.CSIHnEDLE bit to 1.

EDL function works as follows:

- The data has to be broken into 16-bit blocks plus remainder. For example, data of 42 bits would be broken into two 16-bit blocks plus 10 bits.
- The bit length of the remainder is set as “data length” in CSIHnCFGx.CSIHnDLSx[3:0].
- For transmitting the 16-bit blocks, CSIHnTX0W.CSIHnEDL must be set to 1. In this case, the data written to CSIHnTX0W is sent as a 16-bit data length regardless of the CSIHnCFGx.CSIHnDLSx[3:0] bit setting.
- The transfer is complete after a block with the specified data length (the remainder of data specified with CSIHnTX0W.CSIHnEDL = 0) has been sent.

#### Example

Example for sending 40-bit data ($123456789A_H$) to CS0:

40 bits are split into two 16-bit blocks plus 8 bits.

- Initialize CSIHnCFG0.CSIHnDLS0[3:0] = 8.
- To transmit $123456789A_H$ with MSB first, write the following sequence to CSIHnTX0W:
  - $20FE\ 1234_H$ (CSIHnTX0W.CSIHnEDL = 1)
  - $20FE\ 5678_H$ (CSIHnTX0W.CSIHnEDL = 1)
  - $00FE\ 009A_H$ (CSIHnTX0W.CSIHnEDL = 0)

The following figure illustrates the timing.

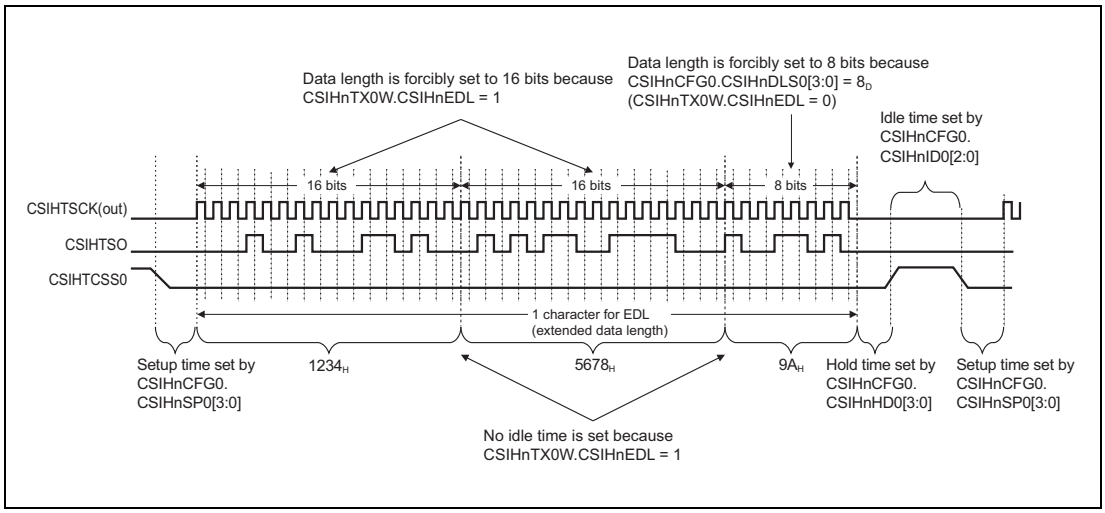



**Figure 17.24 EDL Timing Diagram**

## NOTES

1. 1-bit data length is allowed only when using the EDL mode.
2. If parity is enabled, the parity bit is added after the last bit.
3. When data is sent using extended data length (EDL) function, use the same chip select signal.
4. Example for configuring the data direction:
   - Data to be sent:$123456_H$
   - MSB first:
     Set CSIHnCFGx.CSIHnDIRx = 0
     Write CSIHnTX0W = 20FE $1234_H$ (EDL bit = 1)
     Write CSIHnTX0W = 00FE $0056_H$ (EDL bit = 0)
   - LSB first:
     Set CSIHnCFGx.CSIHnDIRx = 1
     Write CSIHnTX0W = 20FE $3456_H$ (EDL bit = 1)
     Write CSIHnTX0W = 00FE $0012_H$ (EDL bit = 0)
5. Operation is not guaranteed if CSIHnTX0W.CSIHnEOJ and CSIHnTX0W.CSIHnEDL are simultaneously set to “1” while CSIHnCTL1.CSIHnJE = 1 and CSIHnCTL1.CSIHnEDLE = 1.
6. EDL mode cannot be used in receive-only mode of slave mode. (CSIHnCTL2.CSIHnPRS[2:0] = $111_B$, CSIHnCTL0.CSIHnTXE = 0, CSIHnCTL0.CSIHnRXE = 1)

### 17.5.9 Serial Data Direction Selection

The serial data direction is selectable for each chip select signal using the CSIHnDIRx bit in the CSIHnCFGx register.

The examples below show communication for a data length of 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).

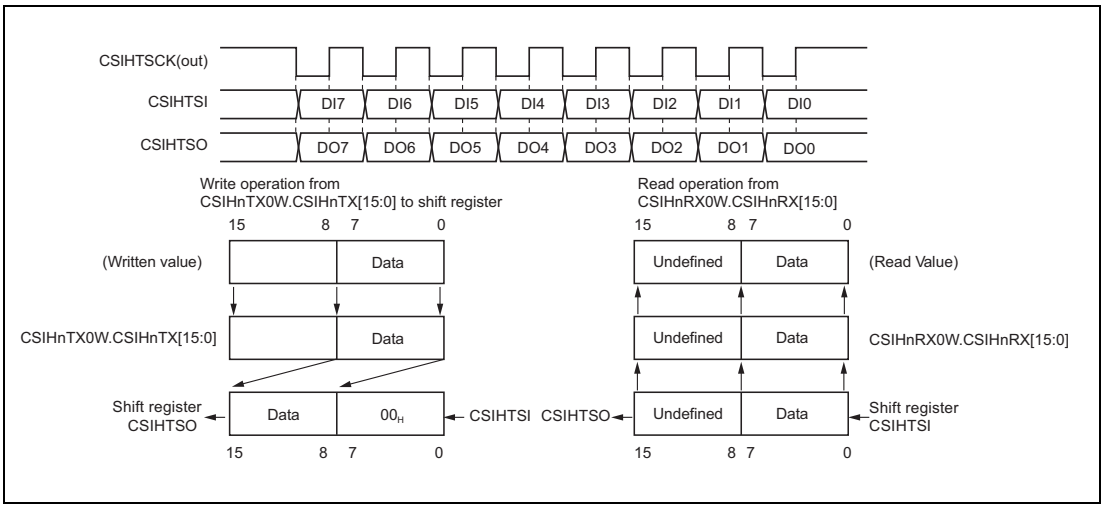



**Figure 17.25 Serial Data Direction Select Function - MSB First (CSIHnDIRx = 0)**

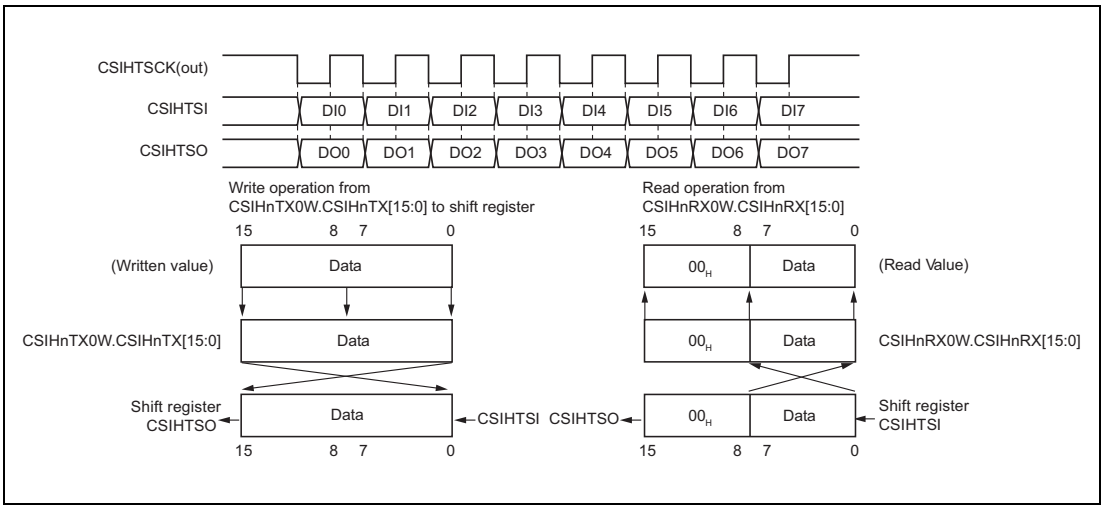



**Figure 17.26 Serial Data Direction Select Function - LSB First (CSIHnDIRx = 1)**

### 17.5.10 Slave Select (SS) Function

The Slave Select (SS) function enables communication between one master and multiple slaves (SPI communications).

In master mode, the master device outputs the slave select signal (CSIHTCSSx) to a slave. Communication by a device in slave mode is enabled when the slave input select signal (CSIHTSSI) is at the low level.

See the **Section 17.5.2, Master/Slave Connections**, for examples of connections using the SS function.

#### 17.5.10.1 Communication Timing Using SS Function

The following figure illustrates the communication signal and timings using the SS function.

In slave mode, the data transfer configuration is determined by the CSIHnCFG0 register.

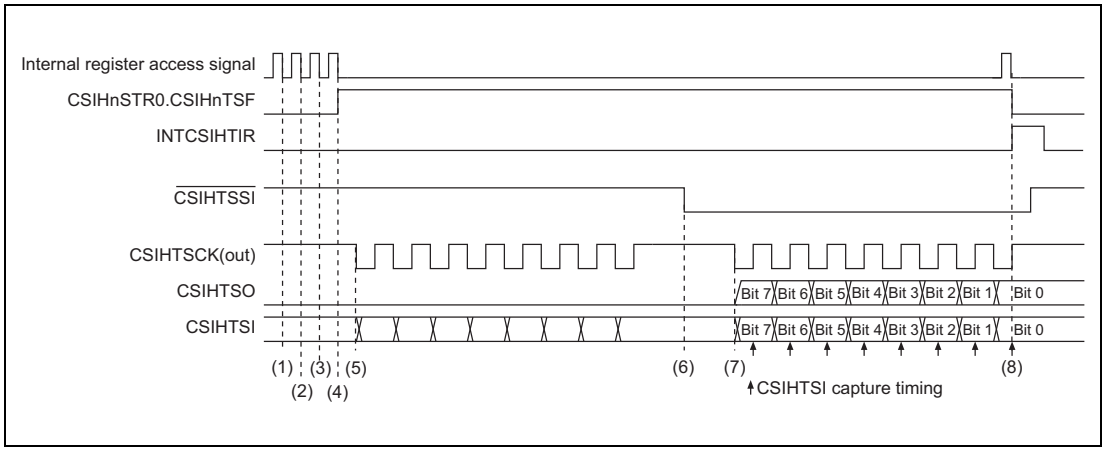



**Figure 17.27 Tx/Rx Timing of Communication Using SS Function**

(1) CSIH enters slave mode by setting CSIHnCTL2.CSIHnPRS[2:0] = $111_B$. CSIHnCFG0.CSIHnCKP0 and CSIHnCFG0.CSIHnDAP0 are 0.

(2) The data length is 8 bits (CSIHnCFG0.CSIHnDLS0[3:0] = $1000_B$). The data direction is MSB first (CSIHnCFG0.CSIHnDIR0 = 0).

(3) The transmit/receive mode is set (CSIHnCTL0.CSIHnTXE = 1, CSIHnCTL0.CSIHnRXE = 1, and CSIHnCTL0.CSIHnPWR = 1). Communication start is permitted.

(4) The transfer status flag CSIHnSTR0.CSIHnTSF is automatically set when transfer data is written to the CSIHnTX0W or CSIHnTX0H transmission register during direct access mode or FIFO mode.

(5) While the signal $\overline{\text{CSIHTSSI}}$ is at high level, transmission/reception is not started, even if an external transmission clock CSIHTSCK is input. Input to CSIHTSI is ignored.

(6) $\overline{\text{CSIHTSSI}}$ falling to low level indicates that CSIHTSO is enabled and ready for transmission.

(7) As soon as the external clock signal CSIHTSCK is detected, the slave transmits data to CSIHTSO and simultaneously captures data from CSIHTSI.

(8) Interrupt INTCSIHTIR indicates that the reception is complete. The CSIHnRX0W/H register can be read.

### 17.5.10.2 CSIHTSSO Operation

| CSIHnPWR | CSIHnTXE | CSIHnRXE | CSIHnSSE | CSIHTSSO |
|---|---|---|---|---|
| 0 | — | — | — | H |
| 1 | — | — | 0 | H |
| | 0 | | 1 | H |
| | 1 | | 1 | Inverse value of $\overline{\text{CSIHTSSI}}$ level |

The CSIHTSSO pin is a signal to control the I/O function of the chip's SO pin when using the SS function.

The CSIHTSO pin is enabled when the CSIHTSSO pin is "High" (the chip's SO pin is being driven).

The CSIHTSO pin is disabled when the CSIHTSSO pin is "Low" (the chip's SO pin is not being driven).

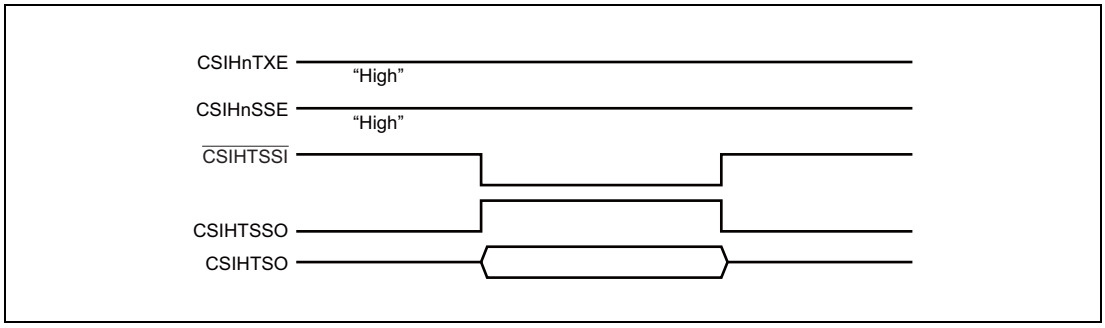



**Figure 17.28 Operation of CSIHTSSO**

**CAUTION**

**If $\overline{\text{CSIHTSSI}}$ pin is changed during communication (CSIHnSTR0.CSIHnTSF = 1), current communication is not guaranteed.**

### 17.5.11 Handshake Function

CSIH features a handshake function to synchronize the master and the slave devices. This function can be enabled/disabled by the CSIHnCTL1.CSIHnHSE bit. For handshake, the signals CSIHTRYI and CSIHTRYO are used.

The busy timing depends on the data phase selection bit CSIHnCFGx.CSIHnDAPx.

#### 17.5.11.1 Slave Mode

If CSIHnCTL1.CSIHnHSE = 1, a low-level CSIHTRYO signal is output when the slave becomes busy. This can happen in two cases:

1. When the next data to be sent is not ready:
   When the slave is in transmit-only mode or transmit/receive mode (CSIHnCTL0.CSIHnTXE = 1) and is in any of the states listed below, the CSIHTRYO outputs the busy state (low level).

**Table 17.43 Memory Mode and Slave Transfer State**

| Memory Mode | Slave Transfer State |
|---|---|
| Direct access mode | When there is no more data to be sent |
| FIFO mode | When there is no more data to be sent (CSIHnSTR0.CSIHnEMF = 1) |
| Dual buffer mode | When CSIHnMCTL2.CSIHnBTST is not set to 1 |
| Transmit-only buffer mode | |

The following examples assume an eight-bit data length.

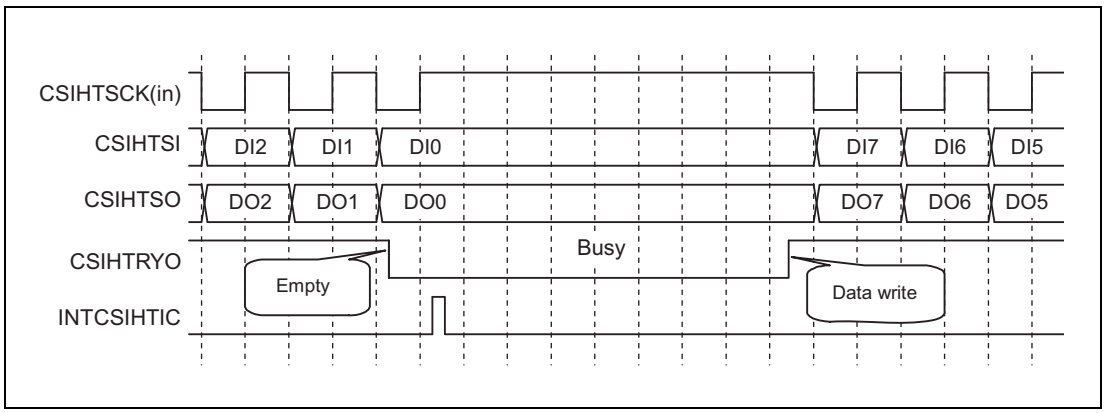



**Figure 17.29 Busy Signal from the Slave (FIFO Mode; CSIHnCFGx.CSIHnDAPx = 0)**

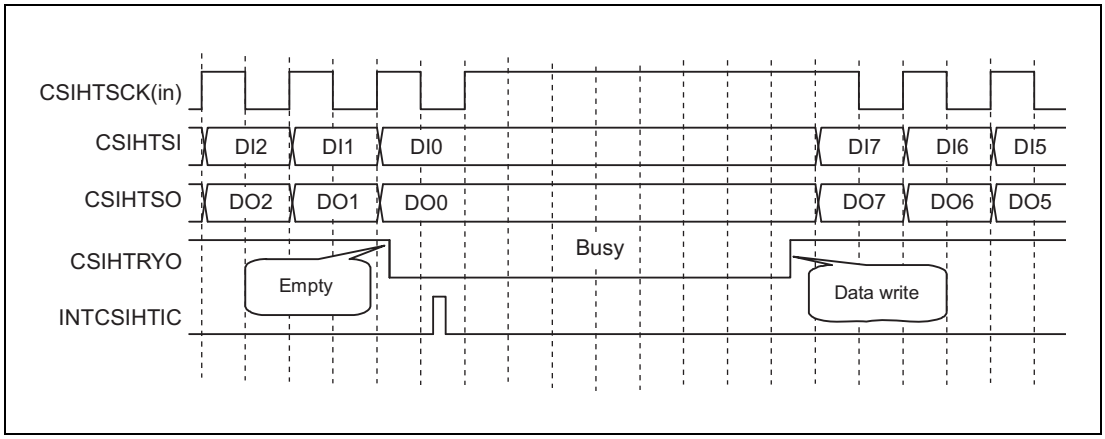



**Figure 17.30 Busy Signal from the Slave (FIFO Mode; CSIHnCFGx.CSIHnDAPx = 1)**

2. When receive register is full:
   When slave is configured in receive-only mode or transmit/receive mode (CSIHnCTL0.CSIHnRXE = 1), and new data cannot be copied from a shift register to CSIHnRX0W/H because the previously received data is still in the CSIHnRX0W/H register (CSIHnRX0W/H is full).
   When CSIHnCTL0.CSIHnRXE is 1 and is in any of the following states, CSIHTRYO outputs busy state (low level).

**Table 17.44 Memory Mode and Slave Reception State**

| Memory Mode | Slave Reception State |
|---|---|
| Direct access mode | When CSIHnRX0W or CSIHnRX0H is full |
| FIFO mode | When received data is remaining in buffer (CSIHnSTR0.CSIHnFLF = 1) |
| Dual buffer mode | No applicable case |
| Transmit-only buffer mode | When CSIHnRX0W or CSIHnRX0H is full |

The following examples assume an eight-bit data length.

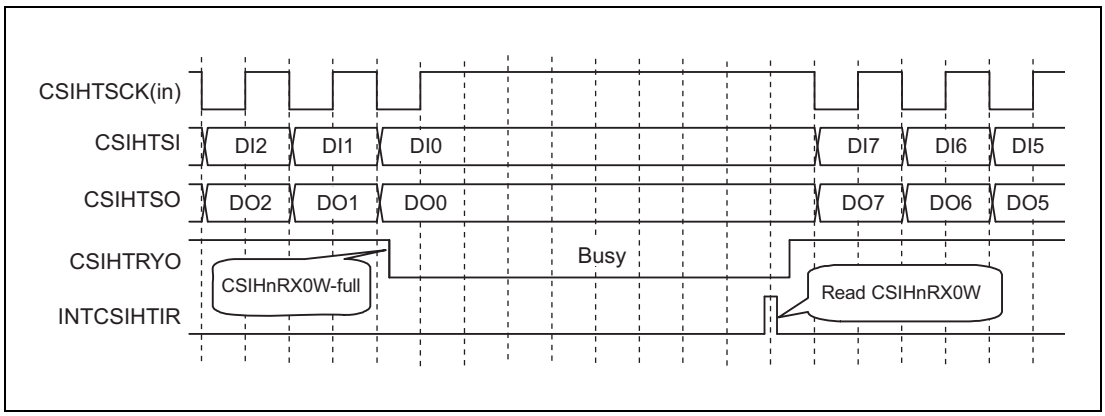



**Figure 17.31 Busy Signal from the Slave (Direct Access Mode; CSIHnCFGx.CSIHnDAPx = 0)**

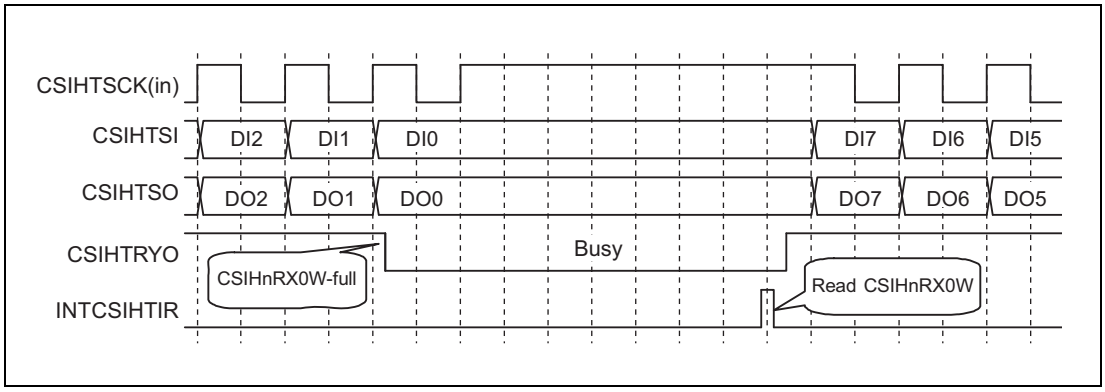



**Figure 17.32 Busy Signal from the Slave (Direct Access Mode; CSIHnCFGx.CSIHnDAPx = 1)**

### 17.5.11.2 Master Mode

When the master detects CSIHTRYI = 0 while CSIHnCTL1.CSIHnHSE = 1, the subsequent transfers are put on hold, the master goes into wait state and suspends the CSIHTSCK clock.

The CSIHTRYI level is checked at each half clock cycle of CSIHTSCK.

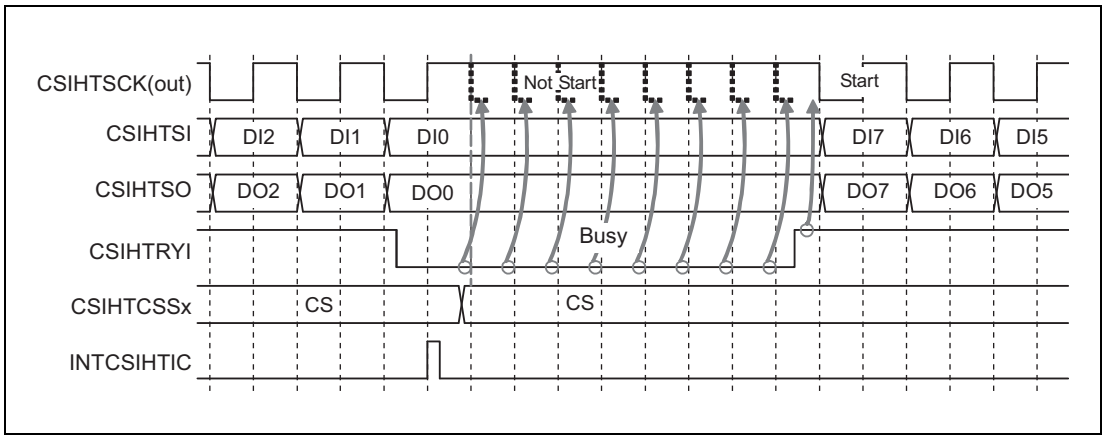



**Figure 17.33 Master's Reaction to CSIHTRYI (CSIHnCFGx.CSIHnDAPx = 0)**

The CSIHTRYI signal must be pulled down by the slave before the next transfer starts. If this is done while data transfer is in progress, the serial clock from the master is suspended after the transfer is complete.

The master resumes the communication as soon as CSIHTRYI becomes high (the slave is "ready").

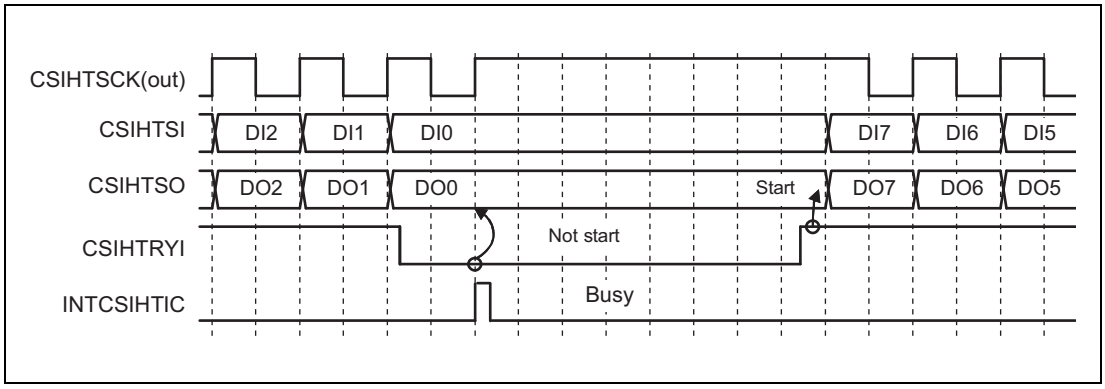



**Figure 17.34 Master's Reaction to CSIHTRYI (CSIHnCFGx.CSIHnDAPx = 1)**

**CAUTIONS**

1. **If multiple slaves are connected, the master must only detect the CSIHTRYI signal of the slave it has selected for communication.**
2. **Even when the CSIHTRYI pin of the master detects a CSIHTRYO signal from the slave during data transfer, the communication is not put on hold but continues until the data transfer is completed.**

### 17.5.12 Error Detection

CSIH can detect five error types:

- Data consistency check error (transmission data)
- Parity error (received data)
- Overrun error (received data)
- Time-out error (in FIFO mode)
- Overflow error (in FIFO mode)

Check for parity, data consistency and time-out errors can be enabled/disabled individually.

If any of these errors is detected, the interrupt request INTCSIHTIRE is generated and the corresponding flags are set.

#### 17.5.12.1 Data Consistency Check

The purpose of the data consistency check is to ensure that the data physically sent as output signal is identical to the original data that was copied to the shift register.

The data consistency check can be enabled/disabled by the CSIHnCTL1.CSIHnDCS bit (when checking data consistency, make sure that PIPCn.PIPCn_m = 1). It will not be enabled if data transmission is disabled (CSIHnCTL0.CSIHnTXE = 0).

When the data consistency check is active, the data transferred from CSIHnTX0W or CSIHnTX0H to the shift register is copied to a separate register. In addition, the physical levels of CSIHTSO are read back via the CSIHTDCS signal into the corresponding shift register.

After completion of the transmission, the sent data is compared with the original transmission data.

Mismatch is considered as a data consistency check error and:

- Interrupt INTCSIHTIRE is generated.
- The CSIHnSTR0.CSIHnDCE bit is set.

Additionally, CSIHnRX0W.CSIHnTDCE of data that contains the error is set.

The data consistency check function is illustrated in the following block diagram.

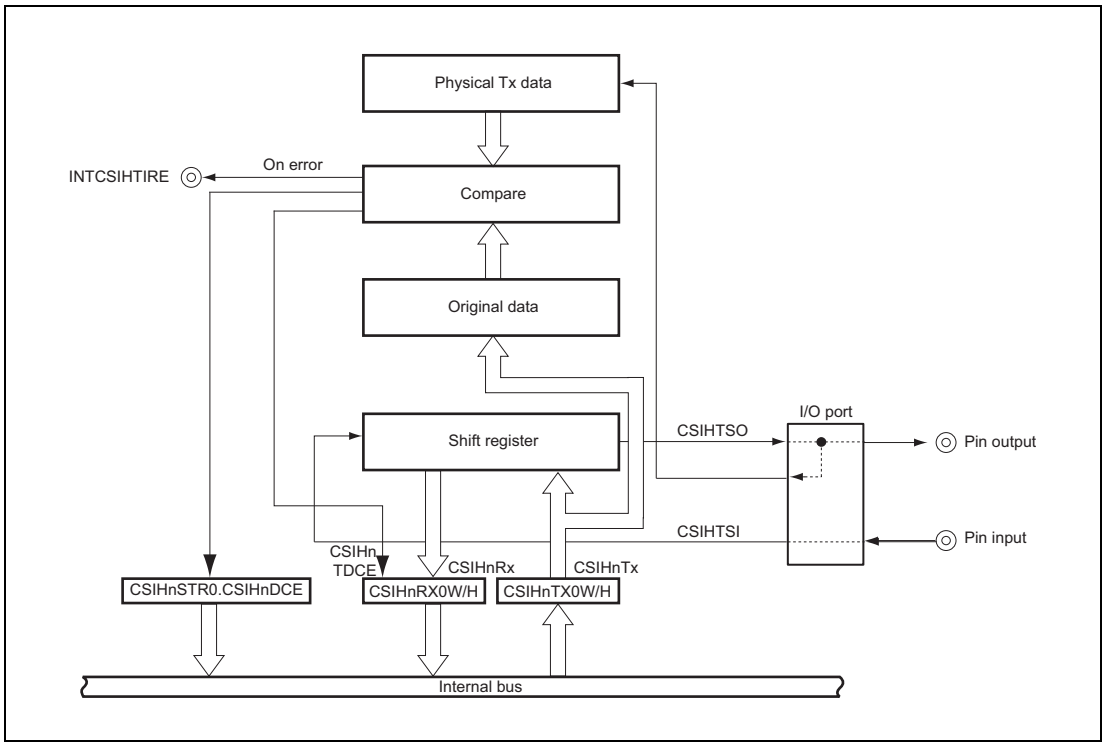



**Figure 17.35 Data Consistency Check Functional Block Diagram**

### 17.5.12.2 Parity Check

CSIH can append a parity bit to the last data bit (even if extended data length is used).

The use and type of parity is specified in CSIHnCFGx.CSIHnPSx[1:0].

Parity check is enabled if CSIHnCFGx.CSIHnPSx[1] = 1.

The parity bit is checked after a reception is complete. In case of parity error:

- Interrupt INTCSIHTIRE is generated.
- The CSIHnSTR0.CSIHnPE bit is set.

Additionally, CSIHnRX0W.CSIHnRPE of data that contains the error is set.

The figure below shows an example.

- Data length is 8 bits.
- The data to be transmitted is $05_H$ and $35_H$.
- Data direction is LSB first.
- Parity type is odd.

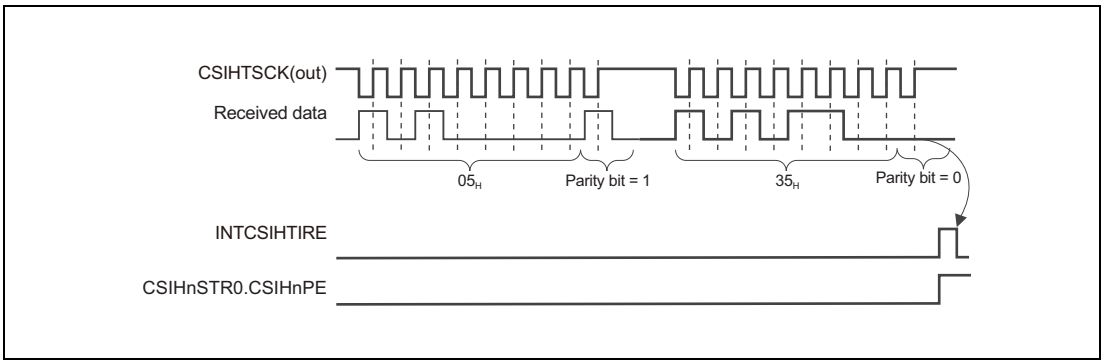



**Figure 17.36 Parity Check Example**

The parity bit of the first data is 1. There is no parity error, because the total number of ones (including the parity bit) is odd.

The parity bit of the second data is 0. This is detected as a parity error, because the total number of ones (including the parity bit) is even.

If the EDL (extended data length) function is used, the parity bit is added after the last bit of the data.

### 17.5.12.3 Time-Out Error

Time-out errors can be checked only in slave FIFO mode.

This error occurs if neither of the following occurred within a certain period of time:

- Received data in FIFO is read
- FIFO receives data from CSIHTSI

The time-out time is defined in CSIHnMCTL0.CSIHnTO[4:0] in units of "8 x transmission clock CSIHTSCK". A time-out error occurs when the specified time is exceeded (the time-out time is not detected when CSIHnMCTL0.CSIHnTO[4:0] = $00000_B$).

The dedicated time-out counter is set by the CSIHnCTL2.CSIHnPRS[2:0] and CSIHnBRSy.CSIHnBRS[11:0] bits.

If the value of the CSIHnBRSy.CSIHnBRS[11:0] bits is left as $000_H$, the dedicated time-out counter does not operate.

The dedicated time-out counter measures the time between the last and the next read operation.

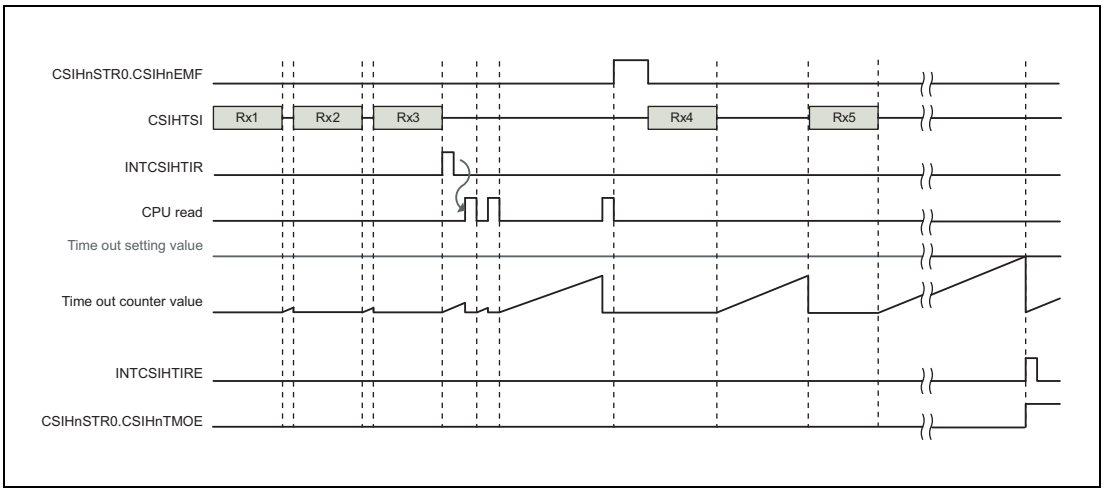



**Figure 17.37 Time-Out Check Functional Timing Diagram**

The start timing of the time-out counter is as follows:

- When reception is completed
- When data read from the CPU is completed
  (The counter does not start if the buffer is empty.)
- When a time-out error is detected

After a time-out error is detected, if data is still available in FIFO, the time-out counter restarts.

If the value set by bits CSIHnMCTL0.CSIHnTO[4:0] is reached again, the INTCSIHTIRE interrupt is output again.

The timeout counter continues to count until received data is read. To stop the counter, read all received data or set CSIHnSTCR0.CSIHnPCT to 1. Note that the pointer is cleared if you perform the latter.

The counter is reset at the following timing:

- Data is read once.
- A new data item is received.

- A timeout error is detected.
- The CSIHnSTCR0.CSIHnPCT bit is set to 1.

If a timeout error occurs, the following occur:

- Interrupt INTCSIHTIRE is generated.
- The CSIHnSTR0.CSIHnTMOE bit is set.

### 17.5.12.4 Overflow Error

An overflow error can occur in FIFO mode. It occurs when transmission data is written to the CSIHnTX0W register while the FIFO buffer is filled with received data.

#### Example

100 data packets have been transmitted. That means, the FIFO contains 100 received data packets. The application starts to read the received data.

While the read operation is in progress, the application begins to write another set of 50 transmission data packets to the FIFO. However, only 10 received data packets have been read up to now, and 90 are still in the FIFO.

In this case, only 38 buffers are available for new transmission data. When the CPU tries to write the 39th data, an overflow error occurs.

This is illustrated in the following figure.

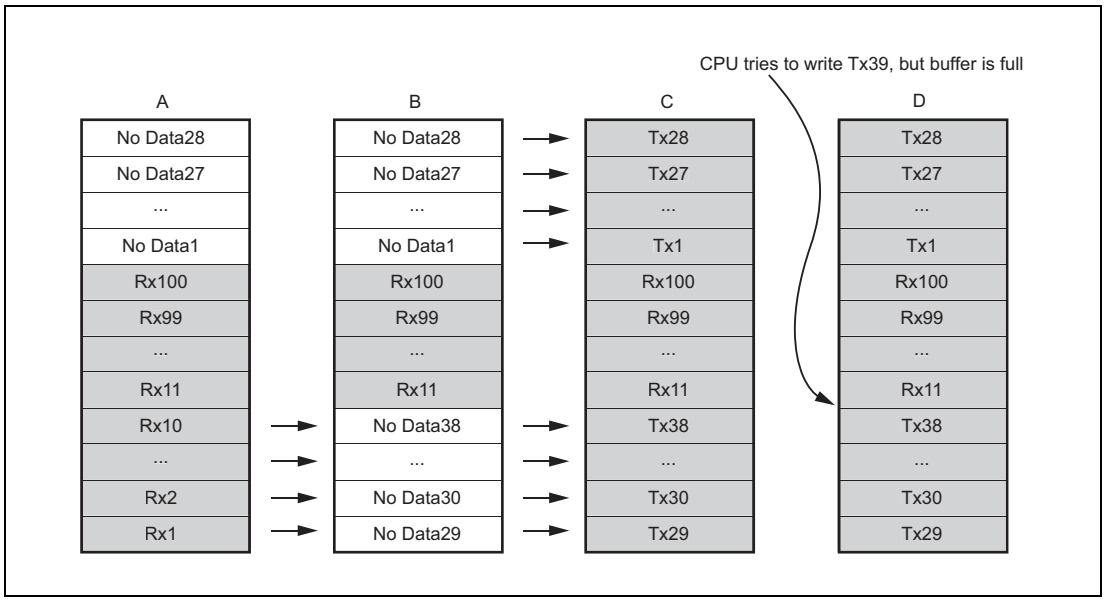



**Figure 17.38 FIFO Overview**

The 39th and subsequent data packets are discarded. The figure below shows the overflow timing.

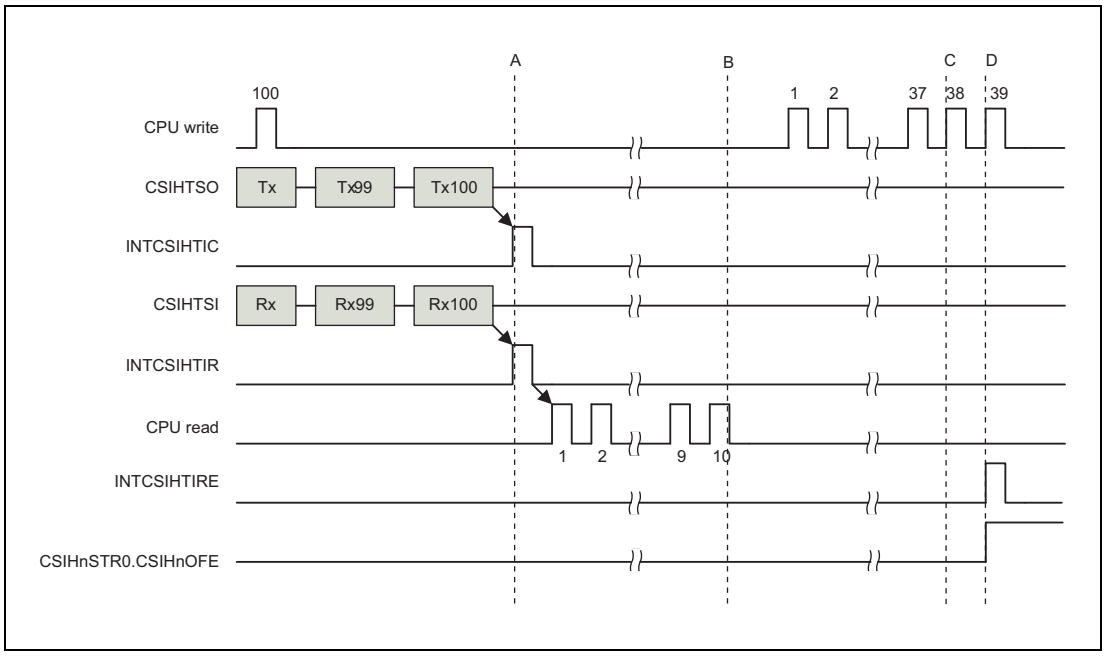



**Figure 17.39 FIFO Overflow Timing**

In case of overflow error:

- Interrupt INTCSIHTIRE is generated.
- The CSIHnSTR0.CSIHnOFE bit is set.

### 17.5.12.5 Overrun Error

An overrun error can occur in direct access, transmit-only buffer, and FIFO modes. It does not occur in dual buffer mode. The overrun error does not occur if data reception is disabled (CSIHnCTL0.CSIHnRXE = 0).

There are two conditions for overrun errors.

#### Condition for errors 1

- In FIFO mode, if the CPU reads the CSIHnRX0W/H register when the number of received data is 0.

#### Condition for errors 2

- In slave mode, when CSIHnCTL1.CSIHnHSE = 0 (handshake function disabled):
  - In direct access mode or transmit-only buffer mode, when reception is completed while the previous received data remains in the CSIHnRX0W/H register.
  - In FIFO mode, when reception is completed while the FIFO buffer is still full of receive data.

#### (1) Direct access/transmit-only buffer

In direct access and transmit-only buffer modes, this error occurs when newly received data cannot be transferred from the shift register to the reception register CSIHnRX0W/H. This happens when CSIHnRX0W/H was not read and therefore contains previously received data.

The following figure illustrates the overrun error detection function.

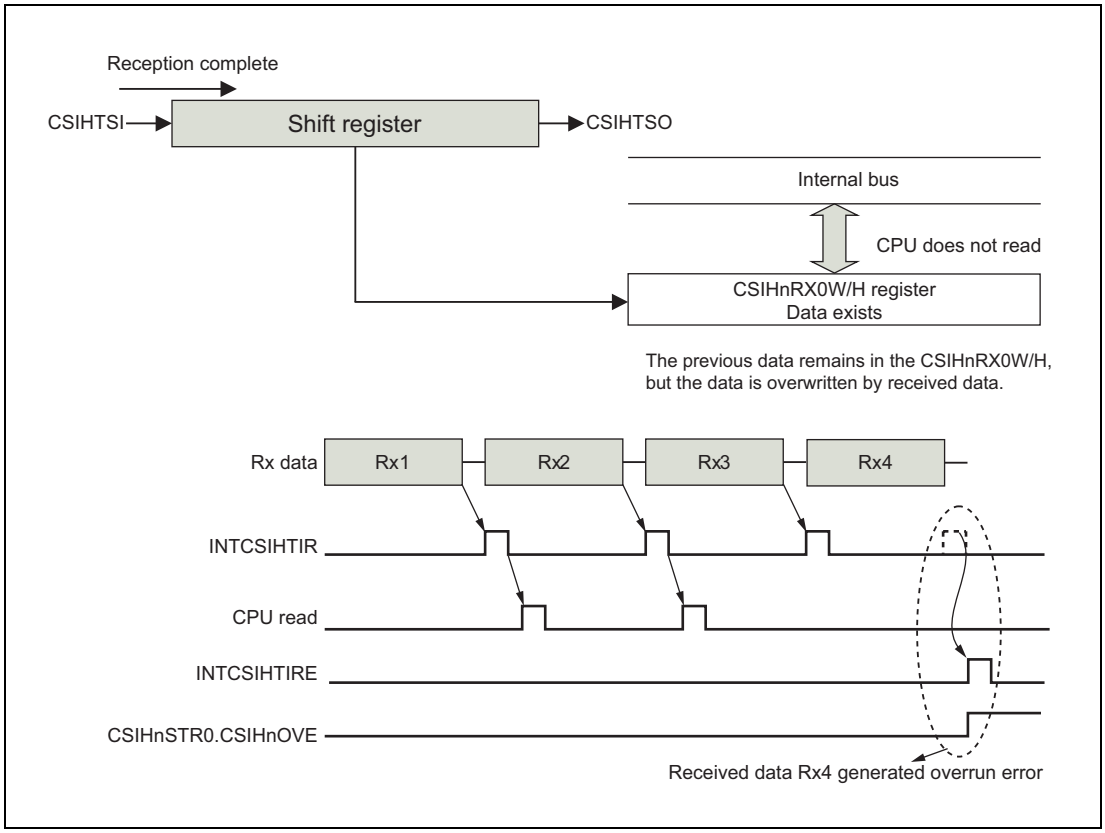



**Figure 17.40 Overrun Error Detection in Direct Access and Transmit-Only Buffer Modes**

**NOTE**

An overrun error can be avoided in slave mode by using handshake function. When handshake is used in slave mode, the receiver (slave) signals to the transmitter (master) that it is busy. The transmitter then waits until the receiver reads its reception register and becomes ready again.

### (2) FIFO mode

In FIFO mode, this error occurs if:

1. Newly received data cannot be transferred from the shift register to the FIFO because the FIFO is full.

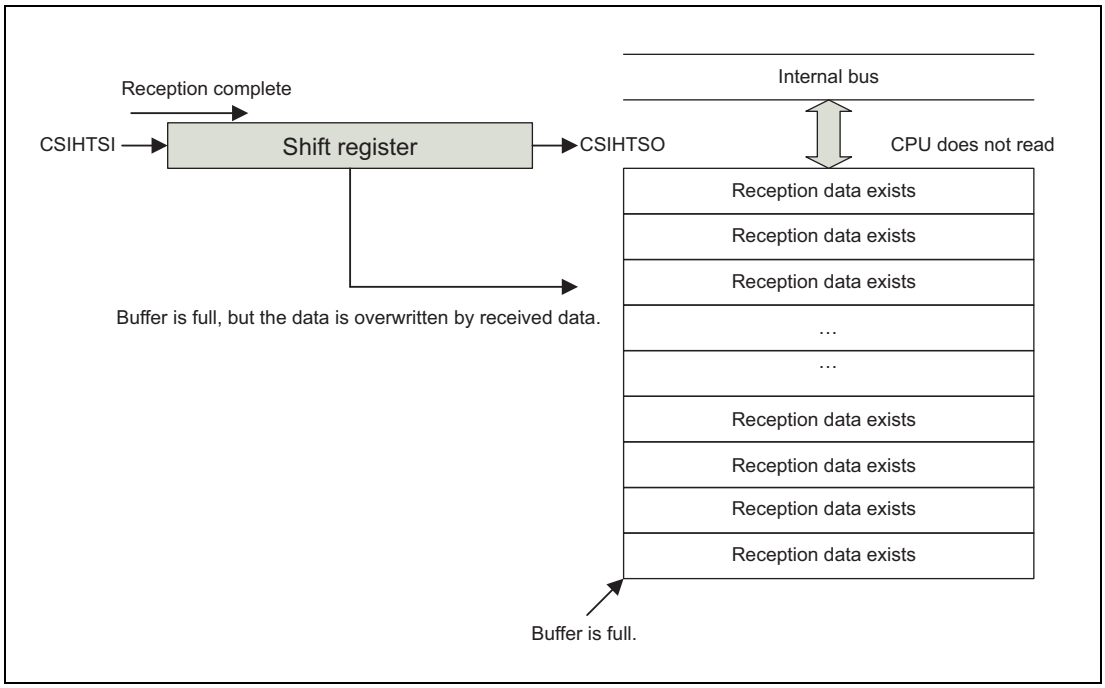



**Figure 17.41 Overrun Error Detection in FIFO Mode (FIFO Full)**

**NOTE**

An overrun error can be avoided in slave mode by using the handshake function. When handshake is used in slave mode, the receiver (slave) signals to the transmitter (master) that it is busy. The transmitter then waits until the receiver reads its reception register and becomes ready again.

2. The CPU attempts to read non-existent receive data.

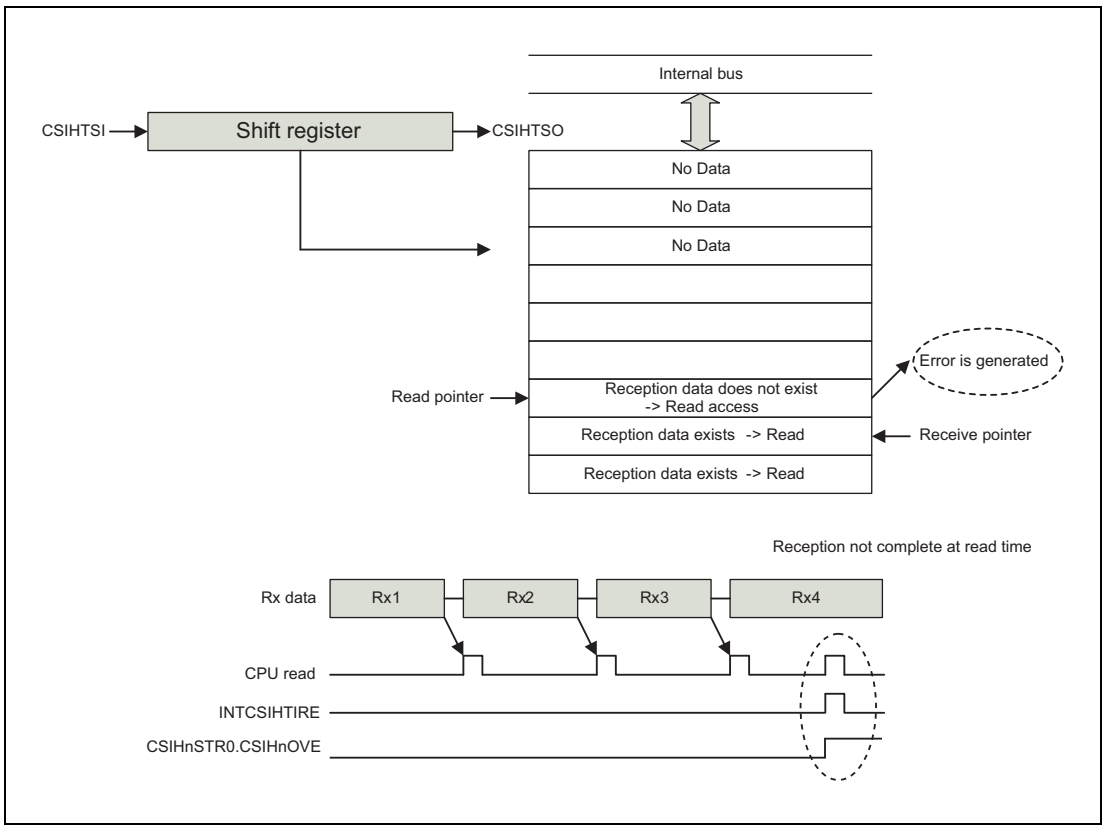



**Figure 17.42 Overrun Error Detection in FIFO Mode (No Data)**

In case of overrun error:

- Interrupt INTCSIHTIRE is generated.
- The CSIHnSTR0.CSIHnOVE bit is set.
- Received data is overwritten and the communication continues.
  (When the CPU tries to read non-existent data, it waits until reception is completed and then resumes reading.)

For details see **Section 17.5.11, Handshake Function**.

### 17.5.13 Loop-Back Mode

Loop-back mode is a special mode for self-test. This feature is only available in master mode.

When this mode is active (CSIHnCTL1.CSIHnLBM = 1), CSIHTCSSx is fixed to the inactive level (the active level is defined by the CSIHnCTL1.CSIHnCSLx value). The transmit and receive signals are internally connected, as shown in the figures below. The signals CSIHTSCK, CSIHTSO, CSIHTSI, and CSIHTCSSx are disconnected from the ports. In addition, the CSIHTSO output level is fixed to low, and CSIHTSCK is set to reset level (High) regardless of the value of the CSIHnCFGx.CSIHnCKPx. The rest of CSIH works as in normal operation.

In order to test CSIH, put it in loop-back mode and carry out normal transfer operations. Then check that the received data is the same as the transmitted data. Any connected device remains unaffected by the loop-back test.

**Table 17.45 Pin Output Level in Loop-Back Mode**

| Pin Name | Output level |
|---|---|
| CSIHTSCK(out) | High level |
| CSIHTCSS[7:0] | Inactive level |
| CSIHTSO | Low level (not dependent on the previous value) |
| Interrupt | Normal function |
| CSIHTRYO | Normal function (Low level) |

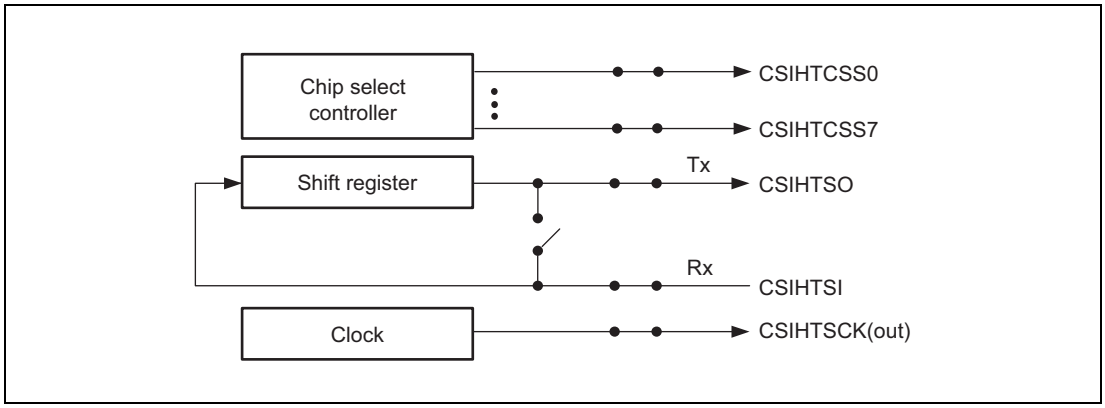



**Figure 17.43 Normal Operation**

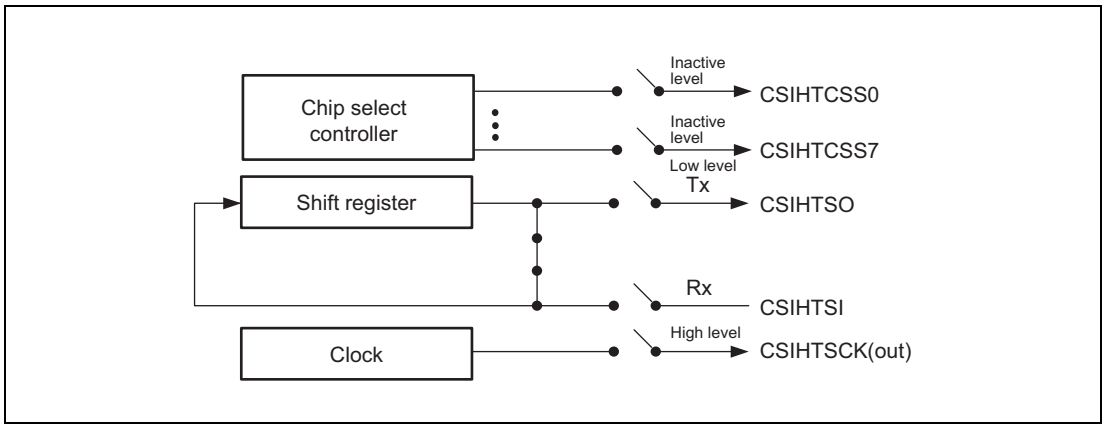



**Figure 17.44 Loop-Back Mode Operation**

### 17.5.14 CPU-Controlled High Priority Communication Function

CSIH has a function to abort low priority communication to perform high priority communication if it receives a high-priority communication request from the CPU while low-priority communication is being used. This function supports transmit-only buffer mode as low priority communication and only direct access mode as high-priority communication. To enable this function, CSIHnCTL1.CSIHnPHE and CSIHnCTL1.CSIHnJE must be set to 1.

The following figure illustrates CPU-controlled high-priority communication.

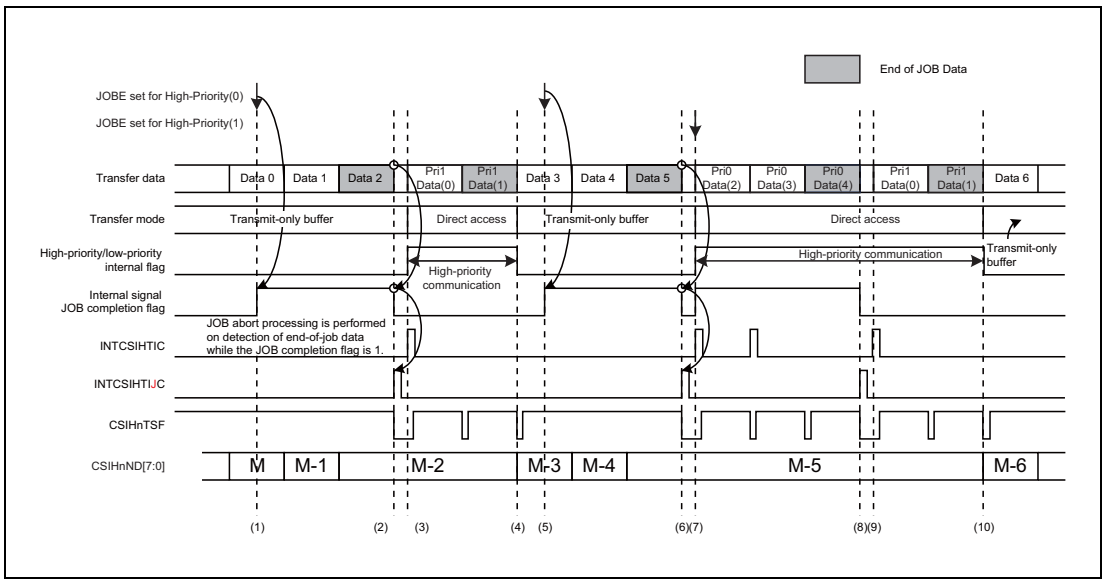



**Figure 17.45 Example of CPU-Controlled High-Priority Communication**

(1) By setting CSIHnCTL0.CSIHnJOBE = 1 during low-priority communication, start of high-priority communication following end-of-job data is notified, and the internal signal flag is set.

(2) When end-of-job data is detected, the current low-priority communication is aborted and the INTCSIHTIJC interrupt occurs. The internal signal job completion flag is cleared due to the communication abort, and memory mode is automatically switched to direct access mode for the subsequent high-priority communication.

(3) The CPU detects the interrupt and starts communication by writing the first transmission data of high-priority communication to CSIHnTX0W or CSIHnTX0H.

(4) When end-of-job data is detected, communication is aborted. At this time, because the internal signal end-of-job flag is set to 0, the CSIH determines that the next communication is low-priority, automatically switches memory mode to transmit-only buffer mode, and then resumes the aborted low-priority communication.

(5) Same as (1) above.

(6) Same as (2) above.

(7) The CPU detects an interrupt and starts communication by writing the first transmission data of high-priority communication to CSIHnTX0W or CSIHnTX0H. The CPU sets CSIHnCTL0.CSIHnJOBE = 1 again to notify that the next communication is high-priority.

(8) When end-of-job data is detected, communication is aborted and the INTCSIHTIJC interrupt is generated. At this time, the CPU determines that the subsequent communication is also high-priority because the internal signal JOB completion flag is 1, and waits for communication to start.

(9) Same as (3) above.

(10) Same as (4) above.

**CAUTION**

**Memory mode is switched automatically when communication is changed from low priority to high priority (from transmit-only buffer mode to direct access mode) and from high priority to low priority (from direct access mode to transmit-only buffer mode).**

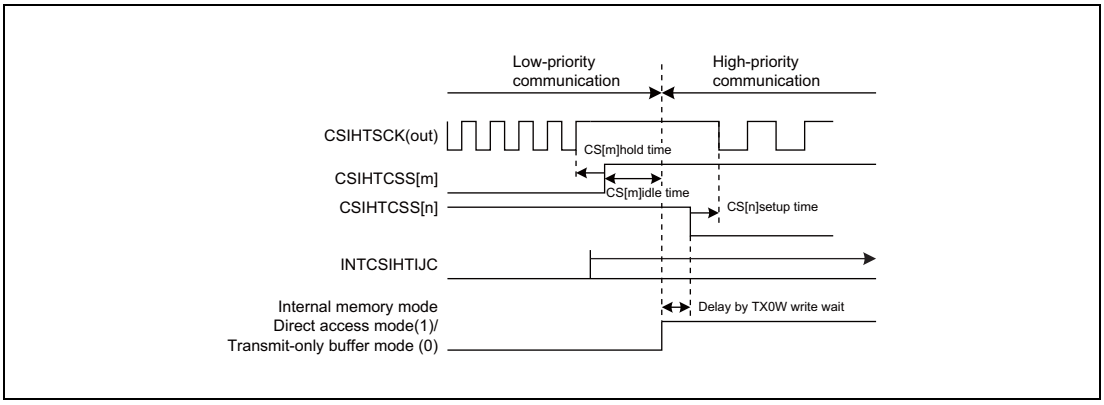



**Figure 17.46 Transition from Low-Priority Mode to High-Priority Mode**

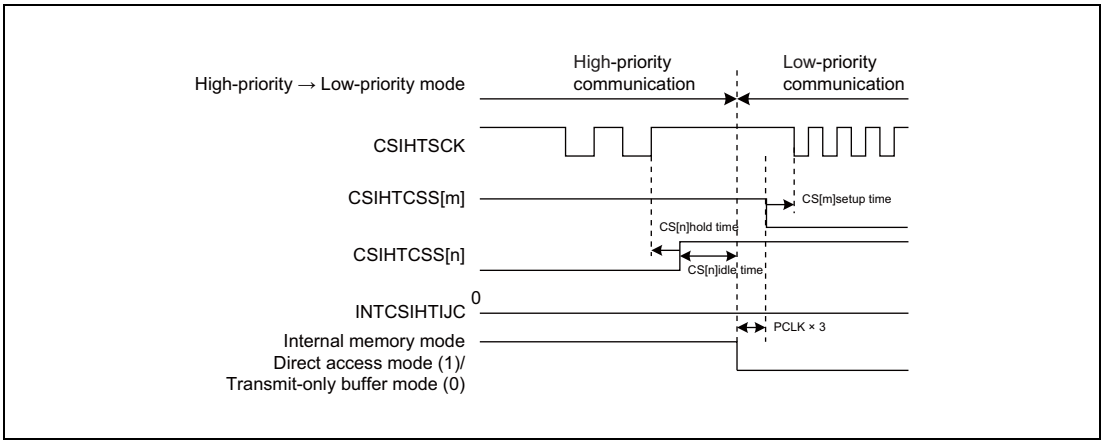



**Figure 17.47 Transition from High-Priority Mode to Low-Priority Mode**

Do not conduct write operation of communication data or CSIHnCTL0.CSIHnJOBE bit operation during setting inhibited period to switch low and high priority communication mode correctly.

CSIHnTX0W register write inhibited period:

- Period from when CSIHnJOBE bit is set for switching to high priority communication mode to when INTCSIHTIJC interrupt is detected.
- Period from when the last data of high priority communication (End of JOB data) is written to when the CSIHnHPST state = 0 is detected.

CSIHnJOBE register write inhibited period:

- Period from when CSIHnJOBE bit is set for switching to high priority communication mode to when INTCSIHTIJC interrupt is detected.

During high communication mode period, there is no setting inhibited period for CSIHnJOBE bit. It is possible to set CSIHnJOBE bit before writing communication data. For example, to communicate multiple JOB data in high priority mode, it is possible to set CSIHnJOBE bit before writing the first communication data.

**CAUTION**

**When CSIHnJOBE bit is set right before the last communication of high priority communication ends, different operations are required depending on the timing at which setting of CSIHnJOBE bit is internally detected.**

**If the setting of the CSIHnJOBE bit is detected before the transfer of the last bit is completed, high priority communication mode continues.**

**When setting of the CSIHnJOBE bit is detected after the transfer of the last bit is completed, the mode temporarily returns to low priority communication. After detection of End of JOB data in low priority communication, the mode changes back to high priority communication.**

### 17.5.15 Enforced Chip Select Idle Setting

It is possible to insert an idle state between two consecutive data transfers by setting CSIHnCFGx.CSIHnIDLx.

1. When CSIHnCFGx.CSIHnIDLx = 0
   If the next CSIHTCSSx is the same as the previous one, an idle state is not inserted, but an inter-data time is inserted instead.
   If the next CSIHTCSSx is different from the previous one, an idle state is inserted.
2. When CSIHnCFGx.CSIHnIDLx = 1
   An idle state is always inserted even if a next CSIHTCSSx is not different from the previous one.

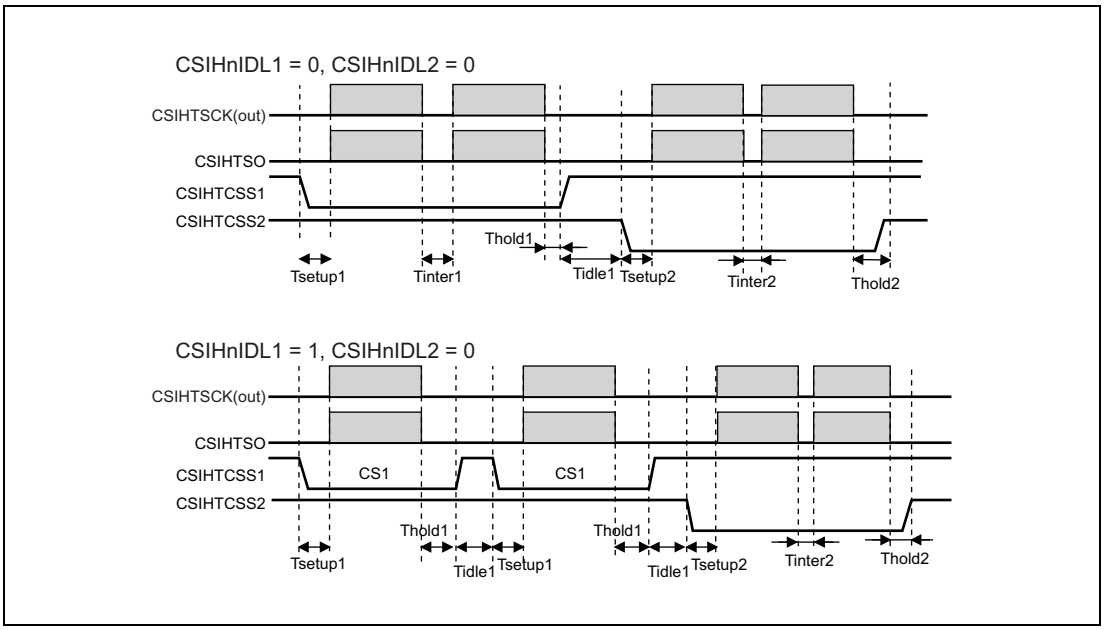



**Figure 17.48 Enforced Chip Select Idle Setting Example**

**CAUTION**

**If the CPU-controlled high priority communication function is enabled (CSIHnCTL1.CSIHnPHE = 1), when the mode is switched from low priority communication to high priority communication or from high priority communication to low priority communication, an IDLE state is inserted regardless of the setting of CSIHnCFGx.CSIHnIDLx bit.**

## 17.6 Operating Procedures

The examples and procedures below are described according to the memory mode in the following order:

- Direct access mode
- Transmit-only buffer mode
- Dual buffer mode
- FIFO mode

### 17.6.1 Procedures in Direct Access Mode

Two examples for a master are provided, one with job mode disabled, and the other with job mode enabled.

#### 17.6.1.1 Transmission/Reception in Master Mode when Job Mode is Disabled

The procedure below assumes the following conditions:

- The transmission data length is 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).
- Transmission direction is MSB first (CSIHnCFGx.CSIHnDIRx = 0).
- Normal clock phase and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- No interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is disabled (CSIHnCTL1.CSIHnJE = 0).
- Normal INTCSIHTIC interrupt timing (CSIHnCTL1.CSIHnSLIT = 0)
- Direct access mode (CSIHnCTL0.CSIHnMBS = 1)

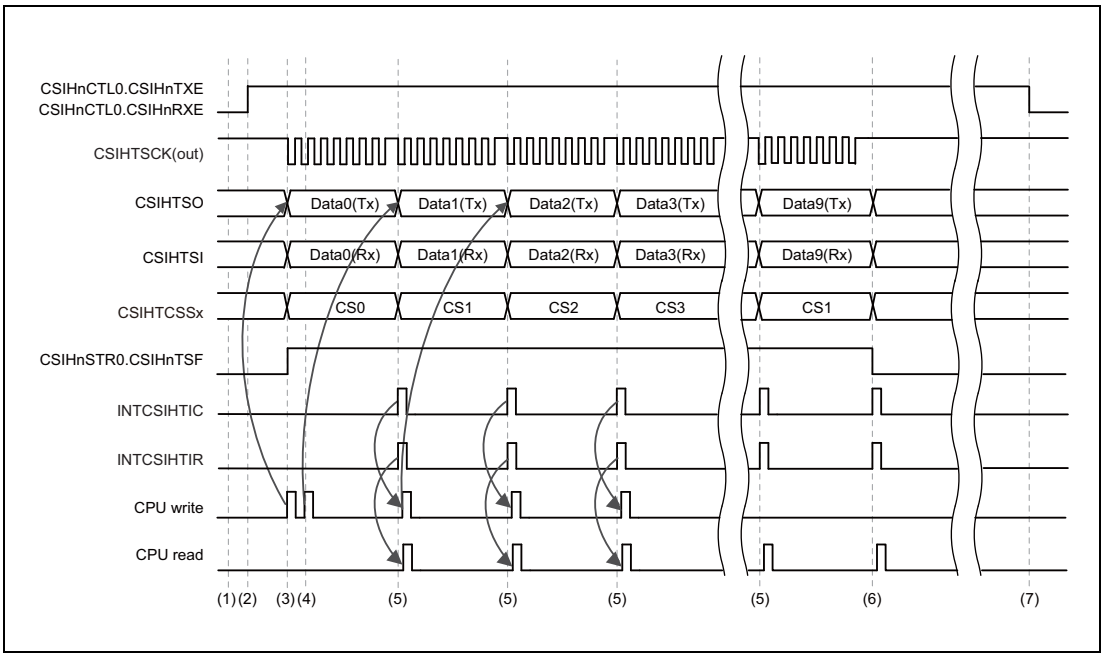



**Figure 17.49 Master in Direct Access Mode, CSIHnCTL1.CSIHnJE = 0**

**Procedure:**

1. Configure the communication protocol in the CSIHnCFGx register. This example uses chip select signals CSIHTCSS0 to CSIHTCSS3.
   Specify the transfer mode and job mode by setting the corresponding bits in registers CSIHnCTL1 and CSIHnCTL2.
2. In the CSIHnCTL0 register, set CSIHnPWR = 1 (enables the clock), CSIHnTXE = 1 (permits transmission), CSIHnRXE = 1 (permits reception), and CSIHnMBS = 1 (selects direct access mode).
3. Write the first data to be sent to the transmission register, CSIHnTX0W. Within the same write operation, activate CS0. Transmission starts automatically when the first data becomes available.
4. Write the second data to CSIHnTX0W. If required, you can change the CS to address a different device. Writing the second data immediately after the first one avoids unnecessary delays between the data.
5. After every data transmission/reception, the interrupt requests INTCSIHTIC and INTCSIHTIR are generated:
   - INTCSIHTIC indicates that the next data can be written to CSIHnTX0W.
   - INTCSIHTIR indicates that the reception register CSIHnRX0W must be read.
6. No more write action is required after completion of data 8. Data 9 (the last data) has been written in advance.
   However, reception register CSIHnRX0W must be read after completion of writing data 8 and 9.
7. Finally, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE to prohibit the transmit/receive operation. When no communication is conducted, set CSIHnCTL0.CSIHnPWR to 0 to minimize power consumption of the CSIHn.

### 17.6.1.2 Transmission/Reception in Master Mode when Job Mode is Enabled

The procedure below is based on the assumption that:

- The transmission data length is 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).
- Transmission direction is MSB first (CSIHnCFGx.CSIHnDIRx = 0).
- Normal clock phase and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- No interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is enabled (CSIHnCTL1.CSIHnJE = 1).
- Normal INTCSIHTIC interrupt timing (CSIHnCTL1.CSIHnSLIT = 0)
- Direct access mode (CSIHnCTL0.CSIHnMBS = 1)
- Two jobs, each transmitting three data.

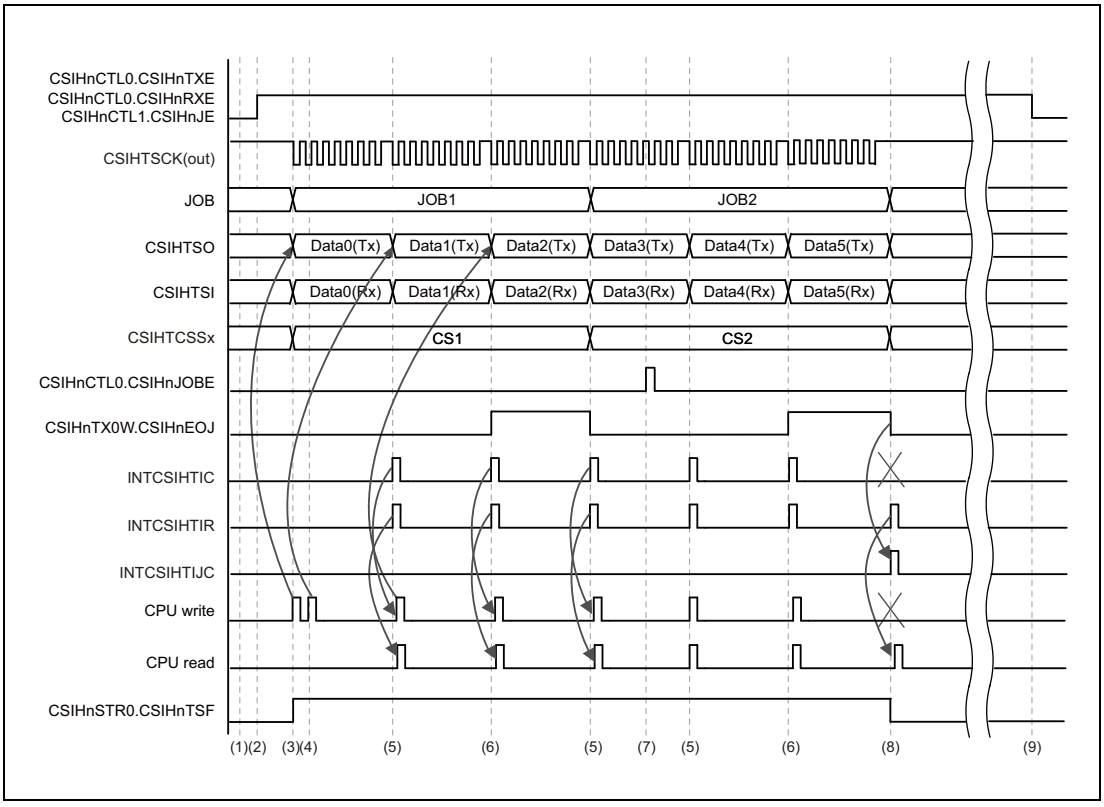



**Figure 17.50 Master in Direct Access Mode, CSIHnCTL1.CSIHnJE = 1**

**Procedure:**

1. Configure the communication protocol in the CSIHnCFGx register. This example uses chip select signals CS1 and CS2.
   Specify the transfer mode and job mode by setting the corresponding bits in registers CSIHnCTL1 and CSIHnCTL2.
2. In the CSIHnCTL0 register, set CSIHnPWR = 1 (enables the clock), CSIHnTXE = 1 (permits transmission), CSIHnRXE = 1 (permits the reception), and CSIHnMBS = 1 (selects direct access mode).
3. Write the first data to be sent to the transmission register CSIHnTX0W. Transmission starts automatically when the first data becomes available.
   The CSIHnSTR0.CSIHnTSF flag indicates that communication is in progress.
4. Write the second data to CSIHnTX0W. Writing the second data immediately after the first one avoids unnecessary delays between the data.
5. After every data transmission/reception, the interrupt requests INTCSIHTIC and INTCSIHTIR are generated.
   - INTCSIHTIC indicates that the next data can be written to CSIHnTX0W.
   - INTCSIHTIR indicates that the reception register, CSIHnRX0W must be read.
6. Setting CSIHnTX0W.CSIHnEOJ = 1 indicates that the last data of the current job is sent. After that, the next job may begin.
7. By setting CSIHnCTL0.CSIHnJOBE = 1, communication is forced to stop at the end of the current job (JOB2).
8. After the forced stop of communication, the interrupt request, INTCSIHTIC is replaced by INTCSIHTIJC. INTCSIHTIR is generated as usual.
   The interrupt request, INTCSIHTIJC indicates that the communication was forcibly stopped at the end of the current job.
   The interrupt request, INTCSIHTIC is not generated. Additionally, the transmission data available in the CSIHnTX0W register is not sent.
9. Finally, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE to prohibit the transmit/receive operation. When no communication is conducted, set CSIHnCTL0.CSIHnPWR to 0 to minimize power consumption of the CSIHn.
   To start another transmission without stopping communication, perform steps 3 and later.

### 17.6.2 Procedures in Transmit-Only Buffer Mode

Two examples for a master is provided, one with job mode disabled, and the other one with job mode enabled.

#### 17.6.2.1 Transmission/Reception in Master Mode when Job Mode is Disabled

The procedure below is based on the assumption that:

- The transmission data length is 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).
- The transmission direction is MSB first (CSIHnCFGx.CSIHnDIRx = 0).
- Normal clock phase and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- No interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is disabled (CSIHnCTL1.CSIHnJE = 0).
- The number of data is 9 (CSIHnMCTL2.CSIHnND[7:0] = $09_H$).
- The transfer start address is $10_H$ (CSIHnMCTL2.CSIHnSOP[6:0] = $10_H$).
- Normal INTCSIHTIC interrupt timing (CSIHnCTL1.CSIHnSLIT = 0)
- Transmit-only buffer mode (CSIHnCTL0.CSIHnMBS = 0,CSIHnMCTL0.CSIHnMMS[1:0] = $10_B$)

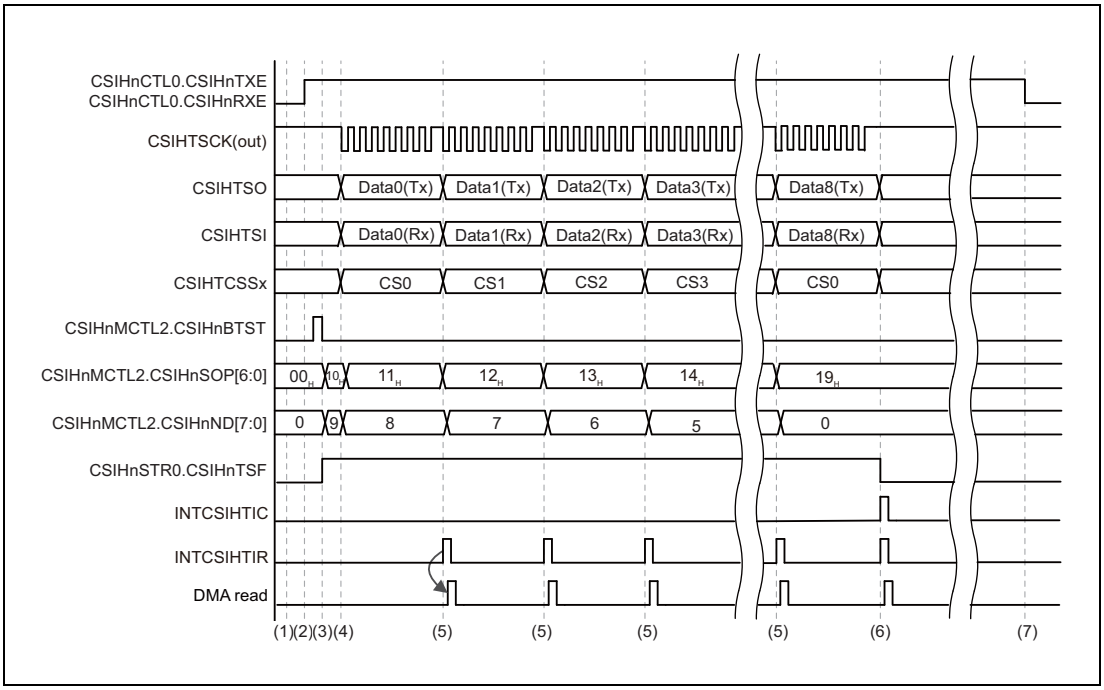



**Figure 17.51 Master in Transmit-Only Buffer Mode, CSIHnCTL1.CSIHnJE = 0**

**NOTE**

The procedure of writing the data into the buffer is not described here.

**Procedure:**

1. Configure the communication protocol in the CSIHnCFGx register. This example uses chip select signals CSIHTCSS0 to CSIHTCSS3.
   Specify the transfer and operating modes by setting the corresponding bits in registers CSIHnCTL1 and CSIHnCTL2.
   Configure the memory mode in CSIHnMCTL0.CSIHnMMS[1:0].
   Set CSIHnMCTL0.CSIHnMMS[1:0] = $10_B$ (transmit-only buffer mode).
2. In the CSIHnCTL0 register, set CSIHnPWR = 1 (enables the clock), CSIHnTXE = 1 (permits transmission), and CSIHnRXE = 1 (permits reception). The CSIHnCTL0.CSIHnMBS bit must be cleared.
3. Configure the transmission pointer and the number of data by setting the CSIHnMCTL2.CSIHnSOP[6:0] and CSIHnMCTL2.CSIHnND[7:0] bits. Start the buffer transfer by setting CSIHnMCTL2.CSIHnBTST.
4. Transmission/reception is started. The CSIHnMCTL2.CSIHnSOP[6:0] bits are automatically incremented and the CSIHnMCTL2.CSIHnND[7:0] bits are decremented after each data transmission.
5. After every data reception, the interrupt request, INTCSIHTIR is generated. INTCSIHTIR indicates that the receive register CSIHnRX0W must be read.
6. When all transmissions are complete, the interrupt request, INTCSIHTIC is generated.
7. Finally, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE to prohibit the transmit/receive operation. When no communication is conducted, set CSIHnCTL0.CSIHnPWR to 0 to minimize power consumption of the CSIHn.

### 17.6.2.2 Transmission/Reception in Master Mode when Job Mode is Enabled

The procedure below is based on the assumption that:

- The transmission data length is 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).
- The transmission direction is MSB first (CSIHnCFGx.CSIHnDIRx = 0).
- Normal clock phase and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- No interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is enabled (CSIHnCTL1.CSIHnJE = 1).
- The number of data is 8 (CSIHnMCTL2.CSIHnND[7:0] = $08_H$).
- The transfer start address is $10_H$ (CSIHnMCTL2.CSIHnSOP[6:0] = $10_H$).
- Normal INTCSIHTIC interrupt timing (CSIHnCTL1.CSIHnSLIT = 0)
- Transmit-only buffer mode (CSIHnCTL0.CSIHnMBS = 0,CSIHnMCTL0.CSIHnMMS[1:0] = $10_B$)

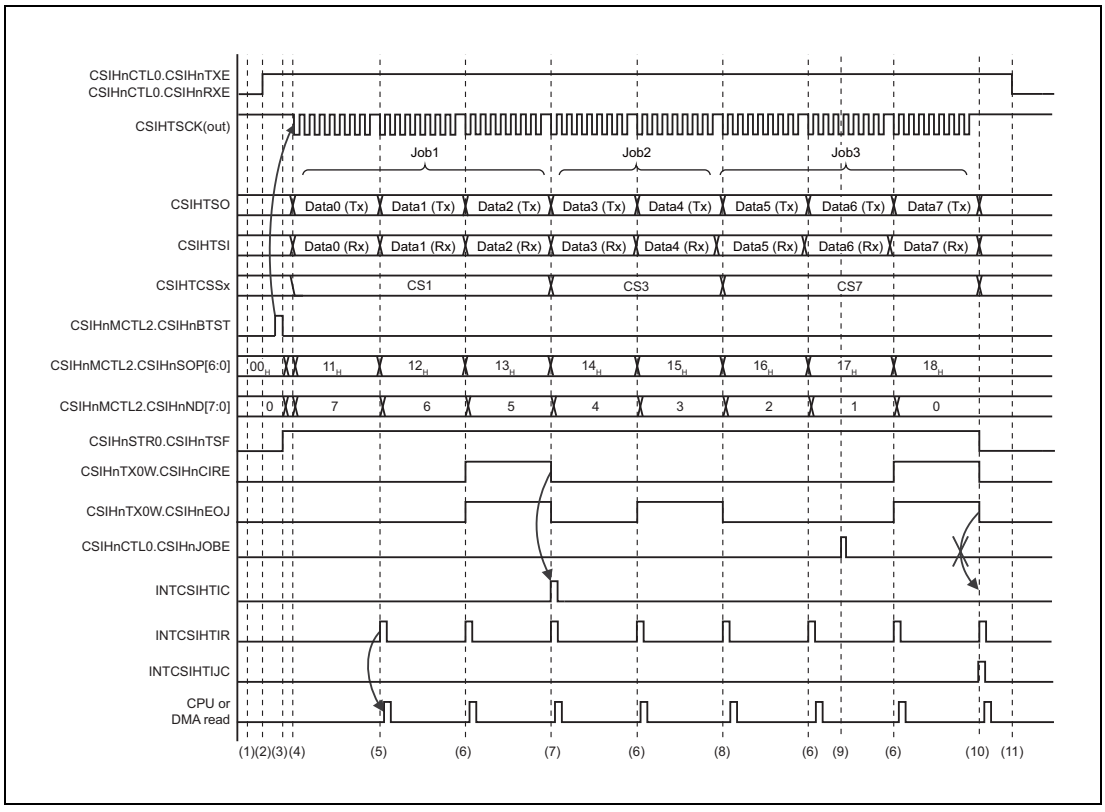



**Figure 17.52 Master in Transmit-Only Buffer Mode, CSIHnCTL1.CSIHnJE = 1**

**NOTE**

The process of writing the data into the buffer is not described.

**Procedure:**

1. Configure the communication protocol in the CSIHnCFGx register. This example uses chip select signals CS1, CS3, and CS7.
   Specify the transfer mode and job mode by setting the corresponding bits in registers CSIHnCTL1 and CSIHnCTL2.
   Configure the memory mode in CSIHnMCTL0.CSIHnMMS[1:0].
   Set CSIHnMCTL0.CSIHnMMS[1:0] = $10_B$ (transmit-only buffer mode).
2. In the CSIHnCTL0 register, set CSIHnPWR = 1 (enables the clock), CSIHnTXE = 1 (permits transmission), and CSIHnRXE = 1 (permits reception). The CSIHnCTL0.CSIHnMBS bit must be cleared.
3. Configure the transmission pointer and the number of data by setting the CSIHnMCTL2.CSIHnSOP[6:0] and CSIHnMCTL2.CSIHnND[7:0] bits. Start the buffer transfer by setting CSIHnMCTL2.CSIHnBTST.
4. Transmission is started. The CSIHnMCTL2.CSIHnSOP[6:0] bits are automatically incremented and the CSIHnMCTL2.CSIHnND[7:0] bits are decremented after each data transmission.
5. After every data reception, the interrupt request, INTCSIHTIR is generated. INTCSIHTIR indicates that the receive register CSIHnRX0W must be read.
6. The CSIHnTX0W.CSIHnEOJ = 1 setting indicates that the last data of the current job is sent.
7. The interrupt request INTCSIHTIC is generated. INTCSIHTIC indicates that the last data of the current job (CSIHnTX0W.CSIHnEOJ = 1) was sent with CSIHnTX0W.CSIHnCIRE = 1.
8. The INTCSIHTIC interrupt request is not generated because the last data of the current job (CSIHnTX0W.CHABnEOJ = 1) was sent with CSIHnTX0W.CSIHnCIRE = 0.
9. By setting CSIHnCTL0.CSIHnJOBE = 1, communication is forced to stop at the end of JOB3.
10. After the forced stop of communication, interrupt requests INTCSIHTIJC and INTCSIHTIR are generated at the end of job3.
    The INTCSIHTIJC interrupt request indicates that the communication was forcibly stopped at the end of the current job.
    The INTCSIHTIC interrupt request is not generated because the INTCSIHTIJC interrupt request is generated instead of the INTCSIHTIC interrupt request. Additionally, the transmission data available in the CSIHnTX0W register is not sent.
11. Finally, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE to prohibit the transmit/receive operation. When no communication is conducted, set CSIHnCTL0.CSIHnPWR to 0 to minimize power consumption of the CSIHn.

### 17.6.3 Procedures in Dual Buffer Mode

Examples when job mode is enabled in master mode, disabled in master mode, and disabled in slave mode are provided below.

#### 17.6.3.1 Transmission/Reception in Master Mode when Job Mode is Disabled

The procedure below is based on the assumption that:

- The transmission data length is 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).
- The transmission direction is MSB first (CHABnCFGx.CSIHnDIRx = 0).
- Normal clock phase and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- No interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is disabled (CSIHnCTL1.CSIHnJE = 0).
- The number of data is 9 (CSIHnMCTL2.CSIHnND[7:0] = $09_H$).
- The transfer start address is $10_H$ (CSIHnMCTL2.CSIHnSOP[6:0] = $10_H$).
- Normal INTCSIHTIC interrupt timing (CSIHnCTL1.CSIHnSLIT = 0)
- Dual buffer mode (CSIHnCTL0.CSIHnMBS = 0, CSIHnMCTL0.CSIHnMMS[1:0] = $01_B$)

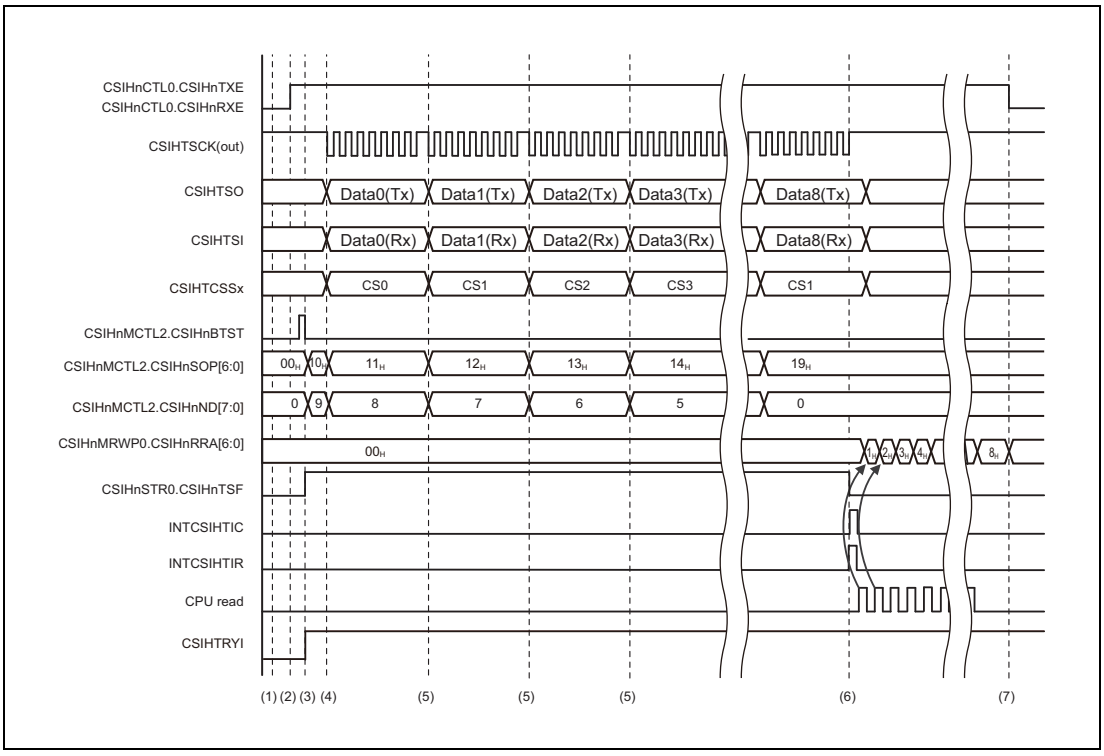



**Figure 17.53 Master in Dual Buffer Mode, CSIHnCTL1.CSIHnJE = 0**

**NOTE**

The process of writing the data into the buffer is not described.

**Procedure:**

1. Configure the communication protocol in the CSIHnCFGx register. This example uses chip select signals CSIHTCSS0 to CSIHTCSS3.
   Specify the transfer and operating modes by setting the corresponding bits in registers CSIHnCTL1 and CSIHnCTL2.
   Configure the memory mode in CSIHnMCTL0.CSIHnMMS[1:0]. Set CSIHnMCTL0.CSIHnMMS[1:0] = $01_B$ (dual buffer mode).
2. In the CSIHnCTL0 register, set CSIHnPWR = 1 (enables the clock), CSIHnTXE = 1 (permits transmission), and CSIHnRXE = 1 (permits the reception). The CSIHnCTL0.CSIHnMBS bit must be cleared.
3. Configure communication by setting the CSIHnMCTL2.CSIHnSOP[6:0] and CSIHnMCTL2.CSIHnND[7:0] bits. Permit buffer transfer by setting CSIHnMCTL2.CSIHnBTST.
4. Transmission is started. The CSIHnMCTL2.CSIHnSOP[6:0] bits are automatically incremented and the CSIHnMCTL2.CSIHnND[7:0] bits are decremented after each data transmission.
5. This is repeated until the last data is transmitted/received.
   The interrupt requests INTCSIHTIC and INTCSIHTIR are not generated.
6. When the last data is transmitted/received, the interrupt requests INTCSIHTIC and INTCSIHTIR are generated.
   The CPU starts to read the received data from the receive buffer. The start address of the read access is specified in CSIHnMRWP0.CSIHnRRA[6:0]. These bits are incremented after each data is read.
7. Finally, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE to prohibit the transmit/receive operation. When no communication is conducted, set CSIHnCTL0.CSIHnPWR to 0 to minimize power consumption of the CSIHn.

### 17.6.3.2 Transmission/Reception in Master Mode when Job Mode is Enabled

The procedure below is based on the assumption that:

- The transmission data length is 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).
- The transmission direction is MSB first (CSIHnCFGx.CSIHnDIRx = 0).
- Normal clock phase and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- No interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is enabled (CSIHnCTL1.CSIHnJE = 1).
- The number of data is 8 (CSIHnMCTL2.CSIHnND[7:0] = $08_H$).
- The transfer start address is $00_H$ (CSIHnMCTL2.CSIHnSOP[6:0] = $00_H$).
- Normal INTCSIHTIC interrupt timing (CSIHnCTL1.CSIHnSLIT = 0)
- Dual buffer mode (CSIHnCTL0.CSIHnMBS = 0, CSIHnMCTL0.CSIHnMMS[1:0] = $01_B$)

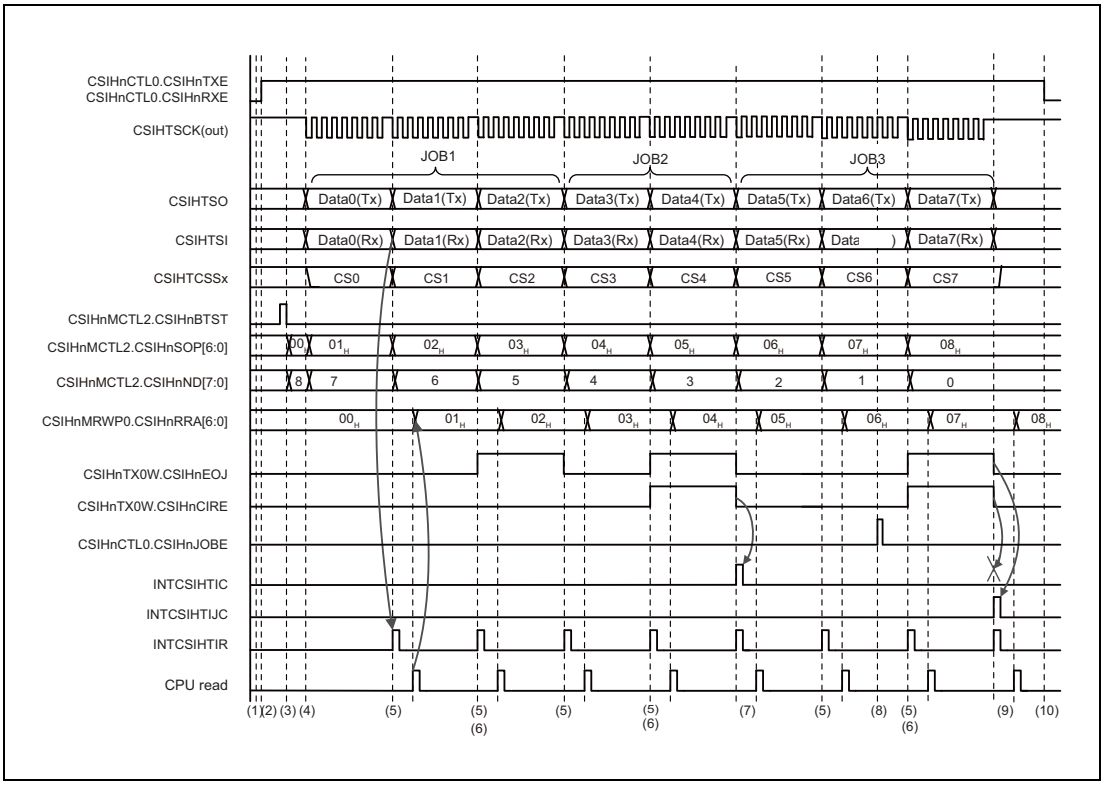



**Figure 17.54 Master in Dual Buffer Mode, CSIHnCTL1.CSIHnJE = 1**

**NOTE**

The process of writing the data into the buffer is not described.

**Procedure:**

1. Configure the communication protocol in the CSIHnCFGx register. This example uses chip select signals CS0 to CS7.
   Specify the transfer and operating modes by setting the corresponding bits in registers CSIHnCTL1 and CSIHnCTL2.
   Configure the memory mode in CSIHnMCTL0.CSIHnMMS[1:0].
   Set CSIHnMCTL0.CSIHnMMS[1:0] = $01_B$ (dual buffer mode).
2. In the CSIHnCTL0 register, set CSIHnPWR = 1 (enables the clock), CSIHnTXE = 1 (permits transmission), and CSIHnRXE = 1 (permits reception). The CSIHnCTL0.CSIHnMBS bit must be cleared.
3. Configure communication by setting the CSIHnMCTL2.CSIHnSOP[6:0] and CSIHnMCTL2.CSIHnND[7:0] bits. Start the buffer transfer by setting CSIHnMCTL2.CSIHnBTST.
4. Transmission is started. The CSIHnMCTL2.CSIHnSOP[6:0] bits are automatically incremented, and the CSIHnMCTL2.CSIHnND[7:0] bits are decremented after each data transmission. This is repeated until the last data is transmitted/received.
5. The INTCSIHTIR interrupt request is generated everytime data is received.
   The INTCSIHTIC interrupt request is not generated because the last data of the current job (CSIHnTX0W.CSIHnEOJ = 1) was sent with CSIHnTX0W.CSIHnCIRE = 0.
6. CSIHnTX0W.CSIHnEOJ = 1 indicates that the last data of the current job is sent.
7. The INTCSIHTIC interrupt request is generated. INTCSIHTIC indicates that the last data of the current job (CSIHnTX0W.CSIHnEOJ = 1) was sent with CSIHnTX0W.CSIHnCIRE = 1.
8. By setting CSIHnCTL0.CSIHnJOBE = 1, communication is forced to stop at the end of JOB3.
9. After the forced stop of communication, interrupt requests INTCSIHTIJC and INTCSIHTIR are generated at the end of JOB3.
   The INTCSIHTIJC interrupt request indicates that the communication was forcibly stopped at the end of the current job.
   The INTCSIHTIC interrupt request is not generated because the INTCSIHTIJC interrupt request is generated instead of the INTCSIHTIC interrupt request. Additionally, the transmission data available in register CSIHnTX0W is not sent.
10. Finally, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE to prohibit the transmit/receive operation. When no communication is conducted, set CSIHnCTL0.CSIHnPWR to 0 to minimize power consumption of the CSIHn.

### 17.6.3.3 Transmit/Receive in Slave Mode when Job Mode is Disabled

The procedure below is based on the assumption that:

- The transmission data length is 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).
- The transmission direction is MSB first (CSIHnCFGx.CSIHnDIRx = 0).
- Normal clock phase and data phase (CSIHnCTL1.CSIHnCKR = 0, CSIHnCFG0.CSIHnDAP0 = 0)
- No interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is disabled (CSIHnCTL1.CSIHnJE = 0).
- The number of data is 9 (CSIHnMCTL2.CSIHnND[7:0] = $09_H$).
- The transfer start address is $10_H$ (CSIHnMCTL2.CSIHnSOP[6:0] = $10_H$).
- Dual buffer mode (CSIHnCTL0.CSIHnMBS = 0, CSIHnMCTL0.CSIHnMMS[1:0] = $01_B$)
- Handshake function is enabled (CSIHnCTL1.CSIHnHSE = 1)

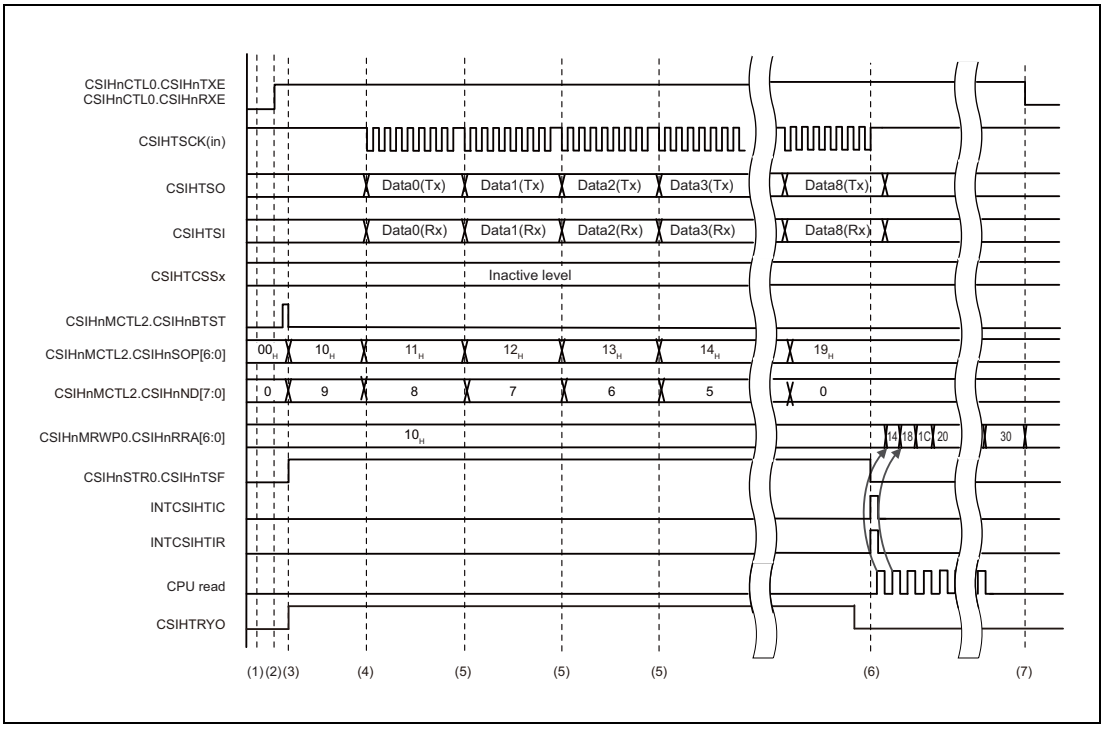



**Figure 17.55 Slave in Dual Buffer Mode, CSIHnCTL1.CSIHnJE = 0**

**NOTE**

The process of writing the data into the buffer is not described.

**Procedure:**

1. Configure the communication protocol in register CSIHnCFG0.
   Specify the transfer and operating modes by setting the corresponding bits in registers CSIHnCTL1 and CSIHnCTL2.
   Configure the memory mode in CSIHnMCTL0.CSIHnMMS[1:0].
   Set CSIHnMCTL0.CSIHnMMS[1:0] = $01_B$ (dual buffer mode).
2. In the CSIHnCTL0 register, set CSIHnPWR = 1 (enables the clock), CSIHnTXE = 1 (permits transmission), and CSIHnRXE = 1 (permits reception). The CSIHnCTL0.CSIHnMBS bit must be cleared.
3. Specify the transfer start address by setting the CSIHnMCTL2.CSIHnSOP[6:0] bits and the number of data by setting the CSIHnMCTL2.CSIHnND[7:0] bits. Permit the buffer transfer by setting CSIHnMCTL2.CSIHnBTST.
4. Transmission is started when the input clock is received from the master. The CSIHnMCTL2.CSIHnSOP[6:0] bits are automatically incremented and the CSIHnMCTL2.CSIHnND[7:0] bits are decremented after each data transmission.
5. This is repeated until the last data is transmitted/received.
   The interrupt requests INTCSIHTIC and INTCSIHTIR are not generated because transmission data is sent from the buffer, and received data is stored in the buffer.
6. When the last data is transmitted/received, the interrupt requests INTCSIHTIC and INTCSIHTIR are generated.
   The CPU starts to read the received data that is stored in the receive buffer. The start address of the read access is specified in CSIHnMRWP0.CSIHnRRA[6:0]. These bits are incremented after each data is read.
7. Finally, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE to prohibit the transmit/receive operation. When no communication is conducted, set CSIHnCTL0.CSIHnPWR to 0 to minimize power consumption of the CSIHn.

### 17.6.4 Procedures in FIFO Mode

Two examples for a master is provided, one with job mode disabled, the other one with job mode enabled.

#### 17.6.4.1 Transmission/Reception in Master Mode when Job Mode is Disabled

The procedure below is based on the assumption that:

- The transmission data length is 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).
- The transmission direction is MSB first (CSIHnCFGx.CSIHnDIRx = 0).
- Normal clock phase and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- No interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is disabled (CSIHnCTL1.CSIHnJE = 0).
- Normal INTCSIHTIC interrupt timing (CSIHnCTL1.CSIHnSLIT = 0)
- FIFO mode (CSIHnCTL0.CSIHnMBS = 0, CSIHnMCTL0.CSIHnMMS[1:0] = $00_B$)

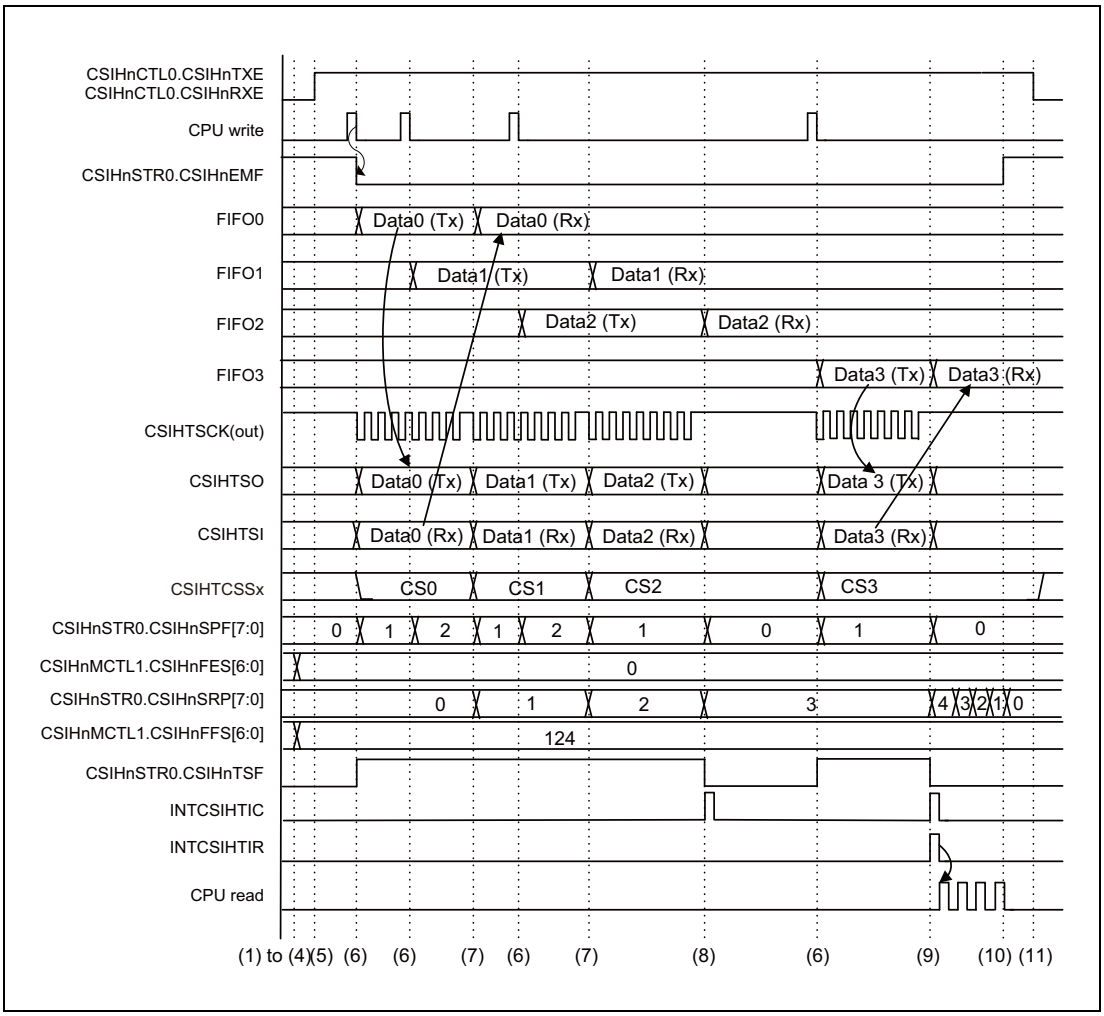



**Figure 17.56 Master in FIFO Mode, CSIHnCTL1.CSIHnJE = 0**

**Procedure:**

1. Configure the communication protocol in the CSIHnCFGx register. Specify the job mode disable and master mode by setting the corresponding bits in registers CSIHnCTL1 and CSIHnCTL2. Specify the FIFO mode by setting CSIHnMCTL0.CSIHnMMS[1:0] = $00_B$. This example uses chip select signals CSIHTCSS0 to CSIHTCSS3.
2. Set CSIHnSTCR0.CSIHnPCT = 1 to clear all buffer pointers.
3. Check that CSIHnSTR0.CSIHnFLF = 0, CSIHnSTR0.CSIHnEMF = 1, and CSIHnSTR0.CSIHnSPF[7:0] = $00_H$.
4. With CSIHnMCTL1.CSIHnFES[6:0], specify the conditions for generating the INTCSIHTIC interrupt output.
   Specify the conditions for generating the INTCSIHTIR interrupt request in the CSIHnMCTL1.CSIHnFFS[6:0] bits.
5. Set CSIHnCTL0.CSIHnPWR = 1 (enables the clock), CSIHnTXE = 1 (permits transmission), and CSIHnRXE = 1 (permits reception). The CSIHnCTL0.CSIHnMBS bit must be cleared.
6. Write the first transmit data to the transmit register CSIHnTX0W. Transmission starts automatically when the first data becomes available.
   Check that CSIHnSTR0.CSIHnEMF = 0.
7. The current transmission is completed. As the CSIHnFES[6:0] bits are not the same as the CSIHnSPF[7:0] bits, the interrupt request INTCSIHTIC is not generated.
8. As the CSIHnFES[6:0] bits are the same as the CSIHnSPF[7:0] bits, the interrupt request INTCSIHTIC is generated.
9. When CSIHnFFS[6:0] = 128 - CSIHnSRP[7:0], the interrupt request INTCSIHTIR is generated. Since CSIHnFES[6:0] = CSIHnSPF[7:0], the interrupt request INTCSIHTIC is generated. After the generation of an interrupt, the CPU starts reading received data that is stored in the receive buffer.
10. When the CPU completes reading the received data that is stored in the receive buffer, CSIHnSTR0.CSIHnEMF is set to 1 and the FIFO buffer becomes empty.
11. Finally, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE to prohibit the transmit/receive operation. In addition, if communication is not performed, set CSIHnCTL0.CSIHnPWR = 0 to minimize the power consumption of CSIHn.

### 17.6.4.2 Transmit/Receive Mode when Job Mode is Enabled in Master Mode

The procedure below is based on the assumption that:

- The transmission data length is 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).
- The transmission direction is MSB first (CSIHnCFGx.CSIHnDIRx = 0)
- Normal clock phase and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- No interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is enabled (CSIHnCTL1.CSIHnJE = 1).
- JOB1 consists of four data, JOB2 consists of three data, and JOB3 consists of five data.
- Normal INTCSIHTIC interrupt timing (CSIHnCTL1.CSIHnSLIT = 0)
- FIFO mode (CSIHnCTL0.CSIHnMBS = 0, CSIHnMCTL0.CSIHnMMS[1:0] = $00_B$)

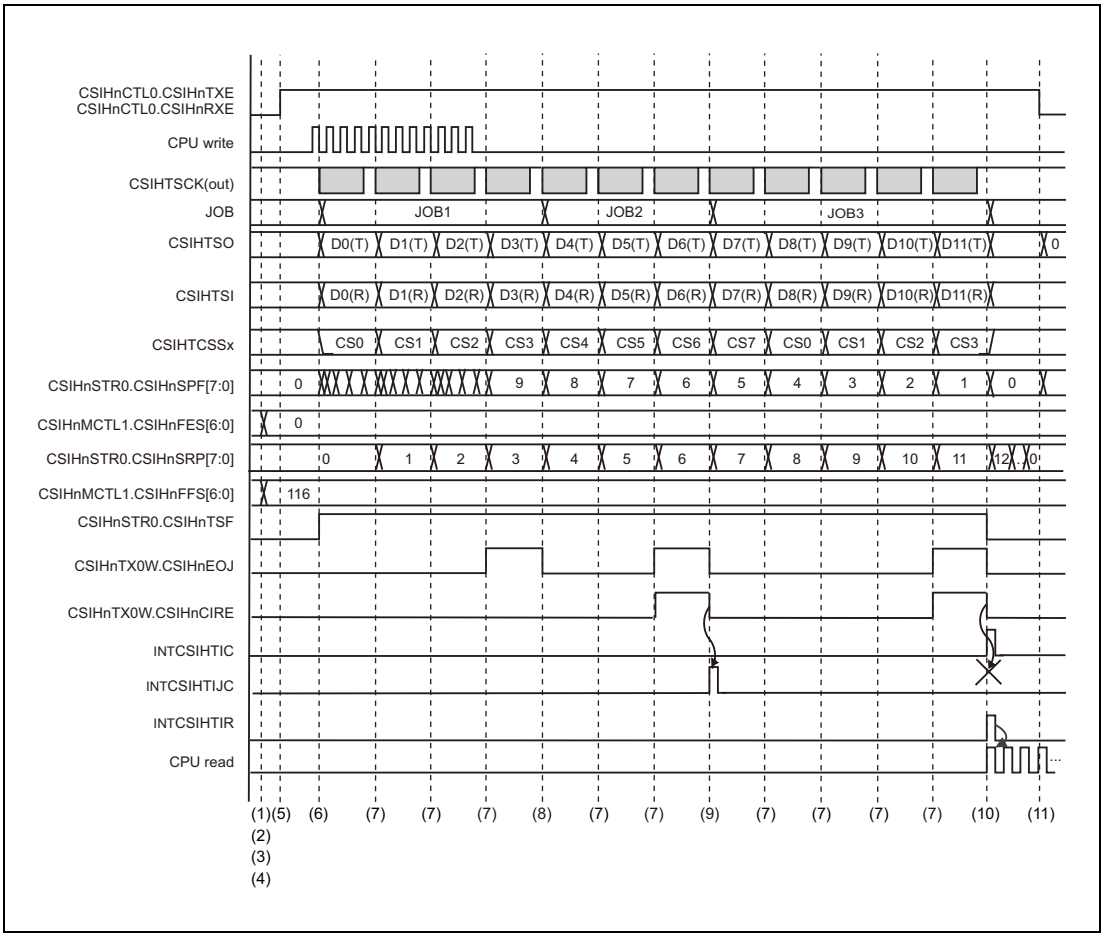



**Figure 17.57 Master in FIFO Mode, CSIHnCTL1.CSIHnJE = 1**

**Procedure:**

1. Configure the communication protocol in the CSIHnCFGx register. Set job mode disable and master mode in the corresponding bits of the CSIHnCTL1 and CSIHnCTL2 registers. Set FIFO mode by setting CSIHnMCTL0.CSIHnMMS[1:0] to $00_B$. This example uses chip select signals CS0 to CS7.
2. Set CSIHnSTCR0.CSIHnPCT to 1 to clear all buffer pointers.
3. Make sure that CSIHnSTR0.CSIHnFLF is set to 0, CSIHnSTR0.CSIHnEMF is set to 1, and CSIHnSTR0.CSIHnSPF[7:0] is set to $00_H$.
4. With CSIHnMCTL1.CSIHnFES[6:0], specify the conditions for generating the INTCSIHTIC interrupt request. With CSIHnMCTL1.CSIHnFFS[6:0], specify the conditions for generating the INTCSIHTIR interrupt request.
5. In the CSIHnCTL0 register, set CSIHnPWR = 1 (enables the clock), CSIHnTXE = 1 (permits transmission), and CSIHnRXE = 1 (permits reception). The CSIHnCTL0.CSIHnMBS bit must be cleared.
6. Write the first data to be sent to the CSIHnTX0W transmission register. Transmission starts automatically when the first data becomes available.
   Make sure CSIHnSTR0.CSIHnEMF is set to 0.
7. The current transmission is completed. Since CSIHnFES[6:0] is not the same as CSIHnSPF[7:0], the interrupt request INTCSIHTIC is not generated.
8. The INTCSIHTIJC interrupt request is not generated because the last data of the current job (CSIHnTX0W.CSIHnEOJ = 1) was sent with CSIHnTX0W.CSIHnCIRE = 0.
9. The INTCSIHTIJC interrupt request is generated because the last data of the current job (CSIHnTX0W.CSIHnEOJ = 1) was sent with CSIHnTX0W.CSIHnCIRE = 1.
10. The INTCSIHTIC interrupt request is generated because CSIHnFES[6:0] = CSIHnSPF[7:0]. INTCSIHTIC is generated so that INTCSIHTIJC is not generated.
    When CSIHnFFS[6:0] = 128 - CSIHnSRP[7:0], the interrupt request INTCSIHTIR is generated. After the generation of the INTCSIHTIR interrupt, CPU starts reading the received data stored in received buffer.
11. Finally, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE to prohibit the transmit/receive operation. When no communication is conducted, set CSIHnCTL0.CSIHnPWR to 0 to minimize power consumption of the CSIHn.

# 17.7 Detection and Correction of Errors in CSIHn RAM

## 17.7.1 ECC for the CSIHn RAM

**Table 17.46** gives an outline of the ECC functions for the CSIH RAM.

**Table 17.46 List of the ECC Functions for the CSIHn RAM**

| Item | Outline of Functions |
|---|---|
| ECC error detection/ correction | The RAM is checked for ECC errors.<br>The following options are selectable.<br>• 2-bit error detection and 1-bit error detection/correction<br>• 2-bit error detection and 1-bit error detection<br>The ECC error detection/correction can be disabled by using through mode.<br>With the initial settings, error detection/correction is enabled. |
| Error notification | When an ECC error is generated, an interrupt signal is generated.<br>• Error notification can be enabled or disabled when an ECC 2-bit error is detected.<br>• Error notification can be enabled or disabled when an ECC 1-bit error is detected.<br>In the initial setting, 2-bit error notification is enabled and 1-bit error notification is disabled.<br>However, if an interrupt is masked by the FEINTFMSK register, an interrupt processing is not executed. |
| Error status | Monitoring for the detection of ECC 2-bit errors and for the detection of ECC1-bit errors is available.<br>A bit for clearing the error status is provided. |
| Address capture | Only one address at which an ECC error has occurred can be captured. A signal is generated upon detection of ECC 2-bit or 1-bit error , and the signal is used as a trigger to capture the error-causing address (when the first (1-bit or 2-bit) error is detected after the flag is cleared). |

## 17.7.2 Interrupt Request

**Table 17.47** lists the ECC interrupt request of CSIHn RAM.

**Table 17.47 CSIHn ECC Interrupt Request (FE-Level Maskable Interrupt)**

| Unit interrupt signal | Description | Name | DMA Trigger Number |
|---|---|---|---|
| INTECCDCSIHn | CSIHn ECC Bit error interrupt | INTECCDCSIH0 | — |
| | | INTECCDCSIH1 | |
| | | INTECCDCSIH2 | |
| | | INTECCDCSIH3 | |

### 17.7.3 Register Base Address

Table 17.48 Register Base Addresses

| Base Address Name | Base Address |
|---|---|
| <ECCCSIH0_base> | FFC7 $0100_H$ |
| <ECCCSIH1_base> | FFC7 $0200_H$ |
| <ECCCSIH2_base> | FFC7 $0300_H$ |
| <ECCCSIH3_base> | FFC7 $0400_H$ |

### 17.7.4 List of Registers

ECC for the CSIHn RAM Registers are listed in the following table.

For details about <ECCCSIHn_base>, see **Section 17.7.3, Register Base Address**.

Table 17.49 List of Registers

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| ECCCSIHn | CSIHn ECC Control Register | ECCCSIHnCTL | <ECCCSIHn_base> + $00_H$ |
| | CSIHn ECC Test Mode Control Register | ECCCSIHnTMC | <ECCCSIHn_base> + $04_H$ |
| | CSIHn ECC Encode/Decode Input/Output Replacement Test Register | ECCCSIHnTED | <ECCCSIHn_base> + $0C_H$ |
| | CSIHn ECC Redundant Bit Data Control Test Register | ECCCSIHnTRC | <ECCCSIHn_base> + $08_H$ |
| | CSIHn ECC Error Address Register 0 | ECCCSIHnAD0 | <ECCCSIHn_base> + $10_H$ |
| | CSIHn ECC Decode Syndrome Data Register | ECCCSIHnSYND | <ECCCSIHn_base> + $0B_H$ |
| | CSIHn ECC 7-Bit Redundant Bit Data Hold Test Register | ECCCSIHnHORD | <ECCCSIHn_base> + $0A_H$ |
| | CSIHn ECC Encode Test Register | ECCCSIHnECRD | <ECCCSIHn_base> + $09_H$ |
| | CSIHn ECC Redundant Bit Input/Output Replacement Register | ECCCSIHnERDB | <ECCCSIHn_base> + $08_H$ |
| SL_READTEST | ECCREAD Test Select Register | SELB_READTEST | FFC7 $8000_H$ |

## 17.7.5 ECCCSIHnCTL — CSIHn ECC Control Register

The ECCCSIHnCTL register controls the mode of the ECC and the status for CSIH.

Bits 7, 5, 4 and 3 should be set (written) while the CSIHn operation is stopped.

In addition, when writing to bit 7, EMCA1 and EMCA0 need to be $01_B$.

**Access:** This register can be read or written in 16-bit units.

**Address:** <ECCCSIHn_base> + $00_H$

**Value after reset:** Undefined

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EMCA1 | EMCA0 | — | — | ECCOVFF | ECER2C | ECER1C | — | ECTHM | — | EC1ECP | EC2EDIC | EC1EDIC | ECER2F | ECER1F | ECEMF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | — |
| R/W | R/W*1 | R/W*1 | R | R | R | R/W*1 | R/W*1 | R | R/W | R | R/W | R/W | R/W | R | R | R |

Note 1. These bits are always read as 0.

**Table 17.50 ECCCSIHnCTL Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | EMCA1 | Access Control Bits 1 and 0 to ECC Mode Selection<br>These bits specify whether updating the ECTHM bit (bit 7) is disabled or enabled. The value written to these bits is not retained. When these bits are read, the read value is always 0. When these bits are $01_B$, writing to bit 7 is enabled. |
| 14 | EMCA0 | |
| 13, 12 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 11 | ECCOVFF | By detecting an error while the error status is set and the new error has anotheraddress than the already latched (not cleared or reset is not issued), this bit is set and error notification is generated.<br><br>0: Overflow is not occurerred afterr reset of clealing ECER2F and ECER1F.<br>1: Error address register overflowed<br><br>**Note:** This bit clear condition is as follows.<br>(1) Reset<br>(2) Writing ECER2C = 1 or ECER1C = 1<br>(3) Selecting through mode enable (ECTHM = 1) |
| 10 | ECER2C | 2-Bit ECC Error Detection Flag Clear<br>This bit clears 2-bit error detection flags of ECER2F (bit 2).<br>This bit is always read as 0. Writing 0 is ignored.<br>Write 1 to this bit while the ECER2F bit is set to clear the ECER2F bit.<br>When a conflict between this bit writing and ECER2F bit setting occurs, writing to this bit has a priority. |
| 9 | ECER1C | 1-Bit ECC Error Detection Correction Accumulation Flag Clear<br>This bit clears 1-bit error detection/correction flags of ECER1F (bit 1). This bit is always read as 0. Writing 0 is ignored.<br>Write 1 to this bit while the ECER1F bit is set to clear the ECER1F bit.<br>When a conflict between this bit writing and ECER1F bit setting occurs, writing to this bit has a priority. |
| 8 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 7 | ECTHM | ECC Function through Mode Selection<br>Set this bit to select whether to enable or disable the ECC function. Setting this bit to 1 disables the ECC function.<br>When writing to this bit, write $01_B$ to the EMCA1 and EMCA0 bits at the same time. Set this bit to 1 to disable the ECC function.<br>0: Passing through mode is disabled (normal operation mode).<br>1: Passing through mode is enabled. (ECC function disable) |

**Table 17.50 ECCCSIHnCTL Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 | EC1ECP | 1-Bit Error Correction Enable<br>This bit specifies whether to enable or disable 1-bit error correction when the ECC error detection/correction is enabled.<br>0: When 1-bit error is detected, the error will be corrected.<br>1: When 1-bit error is detected, the error will not be corrected. |
| 4 | EC2EDIC | 2-Bit Error Detection Interrupt Control<br>This bit controls whether to generate an interrupt when 2-bit error is detected.<br>0: When 2-bit error is detected, an INTECCDCSIHn interrupt will not be generated.<br>1: When 2-bit error is detected, an INTECCDCSIHn interrupt will be generated. |
| 3 | EC1EDIC | 1-Bit Error Detection Interrupt Control<br>This bit controls whether to generate an interrupt when 1-bit error is detected.<br>0: When 1-bit error is detected, an INTECCDCSIHn interrupt will not be generated.<br>1: When 1-bit error is detected, an INTECCDCSIHn interrupt will be generated. |
| 2 | ECER2F | 2-Bit Error Detection Flag<br>This flag indicates whether 2-bit error is detected during read access to the RAM when error determination is enabled (ECTHM = 0). When 2-bit error interrupt is enabled (EC2EDIC = 1) and this flag is set, an ECC 2-bit error interrupt (INTECCDCSIHn) is output.<br>Write 1 to the ECER2C bit (bit 10) to clear the flag. When through mode is enable (ECTHM = 1), this bit is cleared. If 2-bit error is detected again while this bit is set, an interrupt will not be generated.<br>0: 2-bit error has not occurred since this bit was cleared.<br>1: 2-bit error has occurred.<br>This bit is read-only. Writing 0 or 1 does not change internal state. |
| 1 | ECER1F | 1-Bit Error Detection/Correction Flag<br>This flag indicates whether 1-bit error is detected during read access to the RAM when error determination is enabled (ECTHM = 0). Write 1 to the ECER1C bit (bit 9) to clear the flag. When through mode is enabled (ECTHM = 1), this bit is cleared.<br>0: 1-bit error has not occurred since this bit was cleared.<br>1: 1-bit error has occurred.<br>This bit is read-only. Writing 0 or 1 does not change internal state. |
| 0 | ECEMF | ECC Error Message Flag<br>This flag indicates whether an error exists in the current read data bus. This bit is updated whenever the RAM outputs data. This bit is also cleared when through mode is enabled (ECTHM = 1) and there is no 1-bit error in decode circuit input data.<br>0: The current RAM output data does not have bit errors.<br>1: The current RAM output data have bit errors. |

**CAUTION**

Bits 2 and 1 should be cleared when the ECC error message flag (ECEMF) is not set.

## 17.7.6 ECCCSIHnTMC — CSIHn ECC Test Mode Control Register

The ECCCSIHnTMC register is used to switch to the test mode, and this register is for test mode control.

This register can be used when CSIHn is not accessed to RAM.

When writing to bit 7, ETMA1 and ETMA0 need to be $10_B$.

**Access:** This register can be read or written in 16-bit units.

**Address:** <ECCCSIHn_base> + $04_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ETMA1 | ETMA0 | — | — | — | — | — | — | ECTMCE | — | — | ECTRRS | ECREOS | ECENS | ECDCS | ECREIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W*1 | R/W*1 | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W |

Note 1. These bits are always read as 0.

**Table 17.51 ECCCSIHnTMC Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | ETMA1 | Access Control Bits 1 and 0 to ECC Test Mode<br>These two bits specify whether updating the ECTMCE bit (bit 7) is disabled or enabled. The value written to these bits is not retained. When these bits are read, the read value is always 0.<br>When these bits are $10_B$, writing to bits 7 is enabled. |
| 14 | ETMA0 | |
| 13 to 8 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 7 | ECTMCE | ECC Test Mode Enable<br>This bit specifies whether to enable access to test control bits of the test registers and this register. When writing to this bit, write $10_B$ to the ETMA1 and ETMA0 bits at the same time.<br>0: Access to the test mode registers and bits is disabled.<br>1: Access to the test mode registers and bits is enabled.<br><br>Test registers: ECCCSIHnTED, ECCCSIHnTRC, ECCCSIHnSYND, ECCCSIHnHORD, ECCCSIHnECRD, ECCCSIHnERDB<br>Register test control bits: ECTRRS, ECREOS, ECENS, ECDCS, ECREIS |
| 6, 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 | ECTRRS | ECC RAM Read Test Mode Selection<br>This bit selects the targets for reading when the ECCCSIHnTED and ECCCSIHnERDB registers are read. Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously). This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: Read value of the ECCCSIHnTED register will be the write value of the ECCCSIHnTED register. Read value of the ECCCSIHnERDB register will be the write value of the ECCCSIHnERDB register.<br>1: Read value of the ECCCSIHnTED register can read RAM data. Read value of the ECCCSIHnERDB register will be the ECC Data to be written to RAM. |
| 3 | ECREOS | ECC Redundant Bit Output Data Selection<br>This bit specifies which is output to the ECC to be stored in RAM, the ECC data generated for write data or the value of the ECCCSIHnERDB register. Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously).<br>This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: ECC data is generated for write data is stored in RAM.<br>1: The value of ECCCSIHnERDB Register is stored in RAM. |

**Table 17.51 ECCCSIHnTMC Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 2 | ECENS | ECC Encoder Input Selection<br>This bit specifies data written to RAM or the value of the ECCCSIHnTED register as the input to the ECC encoder.<br>Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously).<br>This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: ECC data is generated from write data to RAM<br>1: ECC data is generated from register value of the ECCCSIHnTED. |
| 1 | ECDCS | ECC Decoder Input Selection<br>This bit specifies which data is for generation of syndrome code and error detection, RAM data or the value of ECCCSIHnTED. Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously). This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: Syndrome code generation and error detection are performed from RAM Data.<br>1: Syndrome code generation and error detection are performed from ECCCSIHnTED register value. |
| 0 | ECREIS | ECC Redundant Bit Input Data Selection<br>This bit specifies which ECC data is for generation of syndrome code and error detection, ECC data stored in RAM or the value of the ECCCSIHnERDB . Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously). This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: Syndrome code generation and error detection are performed from ECC data stored in RAM.<br>1: Syndrome code generation and error detection are performed from ECCCSIHnERDB register value. |

### 17.7.7 ECCCSIHnTED — CSIHn ECC Encode/Decode Input/Output Replacement Test Register

In ECC test mode, this register handles test data.

This register value is used to generate ECC data or syndrome code.

This register can be accessed when ECC test mode in enabled (ECCCSIHnTMC.ECTMCE = 1). When ECCCSIHnTMC.ECTMCE = 0, writing to this register is ignored and 0000 $0000_H$ is read.

This register can be used when CSIHn is not accessed to RAM.

**Access:** This register can be read or written in 32-bit units.

**Address:** <ECCCSIHn_base> + $0C_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECEDB[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECEDB[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 17.52 ECCCSIHnTED Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECEDB[31:0] | When ECCCSIHnTMC.ECENS = 1, these bits generate ECC data from value of this register and store the register value in RAM.<br>When ECCCSIHnTMC.ECDCS = 1, these bits generate syndrome code from the value of the register and store the register value in ECC decode syndrome data register (ECCCSIHnSYND).<br>In addition, when ECCCSIHnTMC.ECTRRS = 1, RAM data [31:0] instead of written data is read for the value of this register. |

## 17.7.8 ECCCSIHnTRC — CSIHn ECC Redundant Bit Data Control Test Register

This register is a test register for ECC data in ECC test mode and consists of four 8-bit registers, ECCCSIHnSYND, ECCCSIHnHORD, ECCCSIHnECRD, and ECCCSIHnERDB.

This register can be accessed when ECC test mode is enabled (ECCCSIHnTMC.ECTMCE = 1). When ECCCSIHnTMC.ECTMCE = 0, writing to this register is ignored and 0000 $0000_H$ is read.

This register can be used when CSIHn is not accessed to RAM.

**Access:** This register can be read or written in 32-bit units.

**Address:** <ECCCSIHn_base> + $08_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECCCSIHnSYND (see **Section 17.7.10**) | | | | | | | | ECCCSIHnHORD (see **Section 17.7.11**) | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECCCSIHnECRD (see **Section 17.7.12**) | | | | | | | | ECCCSIHnERDB (see **Section 17.7.13**) | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

## 17.7.9 ECCCSIHnAD0 — CSIHn ECC Error Address Register 0

This is read only register to hold the ECC error occurred address.

When ECC error is detected for permitting ECC error judgment, RAM address is captured by the detected signal as trigger and it is hold as the error occurring address.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <ECCCSIHn_base> + $10_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | ECEAD[30:16] | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECEAD[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 17.53 ECCCSIHnAD0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | When read, the value after reset is returned. |
| 30 to 0 | ECEAD[30:0] | ECEAD0 is a read-only register to hold the address at which an ECC error has occurred.<br>If an ECC error is detected while ECC error detection is enabled, the RAM address is latched using the detection signal as a trigger, and the address is stored in ECEAD0 as the address at which the ECC error has occurred.<br>The address is stored upon detection of the first ECC error while no error status is set. However, if a 1-bit error is followed by a 2-bit error, the address of the lat ter is stored.<br>Only one address can be held in ECEAD0 |

## 17.7.10 ECCCSIHnSYND — CSIHn ECC Decode Syndrome Data Register

This register is a read-only register for storing generated syndrom data in ECC test mode.

Writing to this register is ignored.

This register is read-only when ECC test mode is enabled (ECCCSIHnTMC.ECTMCE = 1). When ECC test mode is disabled (ECCCSIHnTMC.ECTMCE = 0), $00_H$ is read.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <ECCCSIHn_base> + $0B_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | SYND6 | SYND5 | SYND4 | SYND3 | SYND2 | SYND1 | SYND0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 17.54 ECCCSIHnSYND Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. |
| 6 to 0 | SYND[6:0] | These bits store generated syndrome code as needed. |

## 17.7.11 ECCCSIHnHORD — CSIHn ECC 7-Bit Redundant Bit Data Hold Test Register

This register is for storing ECC data for read RAM data in ECC test mode.

Writing to this register is ignored.

This register can be accessed only when ECC test mode is enabled (ECCCSIHnTMC.ECTMCE = 1). When ECC test mode is disabled (ECCCSIHnTMC.ECTMCE = 0), $00_H$ is read.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <ECCCSIHn_base> + $0A_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | HORD6 | HORD5 | HORD4 | HORD3 | HORD2 | HORD1 | HORD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 17.55 ECCCSIHnHORD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. |
| 6 to 0 | HORD[6:0] | These bits store ECC code for read RAM data as needed.<br>When ECCCSIHnTMC.ECTRRS =1 and if ECCCSIHnTED register is read, ECC code is stored. |

## 17.7.12 ECCCSIHnECRD — CSIHn ECC Encode Test Register

This register is a read-only register for storing generated ECC data for read RAM data in ECC test mode.

Writing to this register is ignored.

This register can be accessed only when ECC test mode is enabled (ECCCSIHnTMC.ECTMCE = 1). When ECC test mode is disabled (ECCCSIHnTMC.ECTMCE = 0), $00_H$ is read.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <ECCCSIHn_base> + $09_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | ECRD6 | ECRD5 | ECRD4 | ECRD3 | ECRD2 | ECRD1 | ECRD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 17.56 ECCCSIHnECRD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 6 to 0 | ECRD[6:0] | These bits can read ECC data generated at the time of RAM data writing and can read ECC data for data written in the ECCCSIHnTED register when ECCCSIHnTMC.ECENS =1. |

### 17.7.13 ECCCSIHnERDB — CSIHn ECC Redundant Bit Input/Output Replacement Register

In ECC test mode, this register handles test data.

This register value can be handled as generated ECC data at the time of writing to RAM or as read ECC data at the time of reading RAM data.

This register can be accessed when ECC test mode in enabled (ECCCSIHnTMC.ECTMCE = 1). When ECCCSIHnTMC.ECTMCE = 0, writing to this register is ignored and $00_H$ is read.

**Access:** This register can be read or written in 8-bit units.

**Address:** <ECCCSIHn_base> + $08_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | ERDB6 | ERDB5 | ERDB4 | ERDB3 | ERDB2 | ERDB1 | ERDB0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 17.57 ECCCSIHnERDB Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 6 to 0 | ERDB[6:0] | These bits can store this register value as ECC data when ECCCSIHnTMC.ECREOS = 1. When the register is read while ECCCSIHnTMC.ECREIS =1, the value read from these bits is ECC data read from the RAM.<br>When ECCCSIHnTMC.ECTRRS = 1, ECC data to be stored in RAM will be read for this register value instated of written data. |

## 17.7.14 SELB_READTEST — ECCREAD Test Select Register

SELB_READTEST is used to check read/write access to the CSIHn ECC registers and RS-CANn ECC registers.

Setting 1 to the bit corresponding to each function will enable writing to the read-only bit.

**Access:** This register can be read or written in 16-bit units.

**Address:** FFC7 $8000_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | RTCANFDE7A02 | RTCANFDE7A01 | — | — | — | RTCANE7A1 | — | — | — | — | — | RTCSIHE7A3 | RTCSIHE7A2 | RTCSIHE7A1 | RTCSIHE7A0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R | R | R | R/W | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 17.58 SELB_READTEST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 14 | RTCANFDE7A02 | RS-CAN0 (AFL Buffer) ECC Register Write Access for Testing Purpose Enable/Di sable<br>0: Write access for testing purpose is disabled.<br>1: Write access for testing purpose is enabled (RS-CAN0 ECC read-only bit can be written). |
| 13 | RTCANFDE7A01 | RS-CAN0 (Message Buffer) ECC Register Write Access for Testing Purpose Enable /Disable<br>0: Write access for testing purpose is disabled.<br>1: Write access for testing purpose is enabled (RS-CAN0 ECC read-only bit can be written). |
| 12 to 10 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 9 | RTCANE7A1 | RS-CAN1 ECC Register Write Access for Testing Purpose Enable/Disable<br>0: Wri t e access for testing purpose is disabled.<br>1: Wri t e access for testing purpose is enabled (RS-CAN1 ECC read-onl y bit can be written) . |
| 8 to 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 | RTCSIHE7A3 | CSIH3 ECC Register Write Access for Testing Purpose Enable/Disable<br>0: Write access for testing purpose is disabled.<br>1: Write access for testing purpose is enabled (CSIH3 ECC read-only bit can be written). |
| 2 | RTCSIHE7A2 | CSIH2 ECC Register Write Access for Testing Purpose Enable/Disable<br>0: Write access for testing purpose is disabled.<br>1: Write access for testing purpose is enabled (CSIH2 ECC read-only bit can be written). |
| 1 | RTCSIHE7A1 | CSIH1 ECC Register Write Access for Testing Purpose Enable/Disable<br>0: Write access for testing purpose is disabled.<br>1: Write access for testing purpose is enabled (CSIH1 ECC read-only bit can be written). |
| 0 | RTCSIHE7A0 | CSIH0 ECC Register Write Access for Testing Purpose Enable/Disable<br>0: Write access for testing purpose is disabled.<br>1: Write access for testing purpose is enabled (CSIH0 ECC read-only bit can be written). |

# Section 18 LIN Master Interface (RLIN2)

This section contains a generic description of the LIN master interface (RLIN2).

The first part in this section describes the features specific to RH850/F1K, such as the number of units, register base addresses, etc. The remainder of the section describes the functions and registers of RLIN2.

## 18.1 Features of RH850/F1K RLIN2

### 18.1.1 Number of Units and Channels

This microcontroller has the following number of RLIN2 units and channels.

**Table 18.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of Units | 1 | 2 | 3 |
| Name | RLIN24n (n = 0) | RLIN24n (n = 0, 1) | RLIN24n (n = 0 to 2) |

**Table 18.2 Configurations of RLIN2 Units and Correspondence between the Number of Channels per Unit and the Channel Numbers of the Units**

| Unit Name RLIN24n | Channels per Unit | Unit Channel Number (i) | Channel Name RLIN2m | RH850/F1K 100 pins (3 ch) | RH850/F1K 144 pins (6 ch) | RH850/F1K 176 pins (10 ch) |
|---|---|---|---|---|---|---|
| RLIN240 | 4 | 0 | RLIN20 | √ | √ | √ |
| | | 1 | RLIN21 | √ | √ | √ |
| | | 2 | RLIN22 | √ | √ | √ |
| | | 3 | RLIN23 | | √ | √ |
| RLIN241 | 4 | 0 | RLIN24 | | √ | √ |
| | | 1 | RLIN25 | | √ | √ |
| | | 2 | RLIN26 | | | √ |
| | | 3 | RLIN27 | | | √ |
| RLIN242 | 2 | 0 | RLIN28 | | | √ |
| | | 1 | RLIN29 | | | √ |

**Table 18.3 Indices**

| Index | Description |
|---|---|
| n | Throughout this section, the individual RLIN2 units are identified by the index “n” (n = 0 to 2). |
| m | Throughout this section, the individual channels are identified by the index “m” (m = 0 to 9). |
| i | Throughout this section, the individual channels of units that compose RLIN2 are identified by the index “i” (i = 0 to 3). |
| b | Throughout this section, the individual data buffers implemented in RLIN2 are identified by the index “b” (b = 1 to 8). |

For example, RLN24nGLWBR are the LIN wake-up baud rate select registers, which are the global registers of RLIN2. RLN24nmLiMD are the LIN mode registers, which are the channel registers.

The following lists the indices corresponding to each product.

**Table 18.4 Indices Correspondence of Each Product**

| Indices Correspondence of Each Product | | |
|---|---|---|
| **100 pins** | **144 pins** | **176 pins** |
| i = 0 to 2 (RLIN240) | i = 0 to 3 (RLIN240)<br>i = 0, 1 (RLIN241) | i = 0 to 3 (RLIN240, 1)<br>i = 0,1 (RLIN242) |
| b = 1 to 8 | b = 1 to 8 | b = 1 to 8 |

## 18.1.2 Register Base Addresses

RLIN2 base addresses are listed in the following table.

RLIN2 register addresses are given as offsets from the base addresses.

**Table 18.5 Register Base Addresses**

| Base Address Name | Base Address |
|---|---|
| <RLIN240_base> | FFCE 0000$_H$ |
| <RLIN241_base> | FFCE 0080$_H$ |
| <RLIN242_base> | FFCE 0100$_H$ |

## 18.1.3 Clock Supply

The RLIN2 clock supply is shown in the following table.

**Table 18.6 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| RLIN24n | LIN communication clock sources | CKSCLK_ILIN | Communication clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CKSCLK_ILIN | |

## 18.1.4 Interrupt Request

RLIN2 interrupt requests are listed in the following table.

**Table 18.7 Interrupt Requests (1/2)**

| Unit Interrupt Signal | Description | Interrupt Number |
|---|---|---|
| **RLIN240** | | |
| INTRLIN0 | RLIN20 interrupt | 58 |
| INTRLIN1 | RLIN21 interrupt | 59 |
| INTRLIN2 | RLIN22 interrupt | 162 |
| INTRLIN3 | RLIN23 interrupt | 163 |

Table 18.7 Interrupt Requests (2/2)

| Unit Interrupt Signal | Description | Interrupt Number |
|---|---|---|
| **RLIN241** | | |
| INTRLIN4 | RLIN24 interrupt | 226 |
| INTRLIN5 | RLIN25 interrupt | 227 |
| INTRLIN6 | RLIN26 interrupt | 275 |
| INTRLIN7 | RLIN27 interrupt | 276 |
| **RLIN242** | | |
| INTRLIN8 | RLIN28 interrupt | 285 |
| INTRLIN9 | RLIN29 interrupt | 286 |

### 18.1.5 Reset Sources

RLIN2 reset sources are listed in the following table. RLIN2 is initialized by these reset sources.

Table 18.8 Reset Sources

| Unit Name | Reset Source |
|---|---|
| RLIN24n | All reset sources (ISORES) |

### 18.1.6 External Input/Output Signals

External input/output signals of RLIN2 are listed in the following table.

Table 18.9 External Input/Output Signals

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **RLIN240** | | |
| RLIN2mRX (m = 0 to 3) | RLIN240 receive data input | RLIN2mRX (m = 0 to 3) |
| RLIN2mTX (m = 0 to 3) | RLIN240 transmit data output | RLIN2mTX (m = 0 to 3) |
| **RLIN241** | | |
| RLIN2mRX (m = 4 to 7) | RLIN241 receive data input | RLIN2mRX (m = 4 to 7) |
| RLIN2mTX (m = 4 to 7) | RLIN241 transmit data output | RLIN2mTX (m = 4 to 7) |
| **RLIN242** | | |
| RLIN2mRX (m = 8, 9) | RLIN242 receive data input | RLIN2mRX (m = 8, 9) |
| RLIN2mTX (m = 8, 9) | RLIN242 transmit data output | RLIN2mTX (m = 8, 9) |

# 18.2 Overview

## 18.2.1 Functional Overview

The LIN Master Interface is a hardware LIN communication controller that complies with LIN Specification Package Revision 1.3, 2.0, 2.1, 2.2, and SAEJ2602 (SEP 2005), and automatically performs frame communication and error determination.

**Table 18.10** shows the LIN Master Interface specifications.

**Table 18.10 LIN Master Interface Specifications**

| Item | | Specifications |
|---|---|---|
| Channel count | | 10 channels (In this product, 2-channel version and 4-channel version of RLIN2 is included.) |
| LIN communication function | Protocol | LIN Specification Package Revision 1.3, 2.0, 2.1, 2.2, and SAEJ2602 (SEP 2005) |
| | Variable frame structure | • Transmission break width: 13 to 28 Tbits<br>• Transmission break delimiter width: 1 to 4 Tbits<br>• Inter-byte space (header): 0 to 7 Tbits (space between Sync field and ID field)*1<br>• Response space: 0 to 7 Tbits*1<br>• Inter-byte space: 0 to 3 Tbits (space between data bytes in response area)<br>• Transmit wake-up: 1 to 16 Tbits |
| | Checksum | • Automatic operation for both transmission and reception<br>• Classic or enhanced selectable (for each frame) |
| | Response field data byte count | Variable from 0 to 8 bytes |
| | Frame communication modes | • Mode in which header transmission and response transmission/reception are started with a single transmission start request<br>• Mode in which header transmission and response transmission are started with separate transmission start requests (frame separate mode) |
| | Wake-up transmission and reception | Available in LIN wake-up mode<br>• Wake-up transmission (1 to 16 Tbits)<br>• Wake-up reception<br>Low-level width of input signals measured |
| | Status | • Successful frame/wake-up transmission<br>• Successful header transmission<br>• Successful frame/wake-up reception*2<br>• Successful data 1 reception<br>• Error detection<br>• Operation mode<br>(LIN reset mode, LIN wake-up mode, LIN operation mode, LIN self-test mode) |
| | Error status | • Bit error<br>• Checksum error<br>• Frame timeout error<br>• Physical bus error<br>• Framing error |
| | Baud rate selection | Baud rate conforming to the LIN specifications generated using baud rate generator |
| | Test mode | Self-test mode for user evaluation |
| | Interrupt function | • Successful frame/wake-up transmission<br>• Successful frame/wake-up reception*2<br>• Error detection<br><br>The logical OR of these three events is the interrupt source (INTRLINm) for each channel. |

Note 1. Since the same register is used for configuration, the inter-byte space (header) = response space.
Note 2. For wake-up reception, the input signal low-level width count is indicated.

## 18.2.2 Block Diagram

**Figure 18.1** shows a block diagram of the LIN master interface.

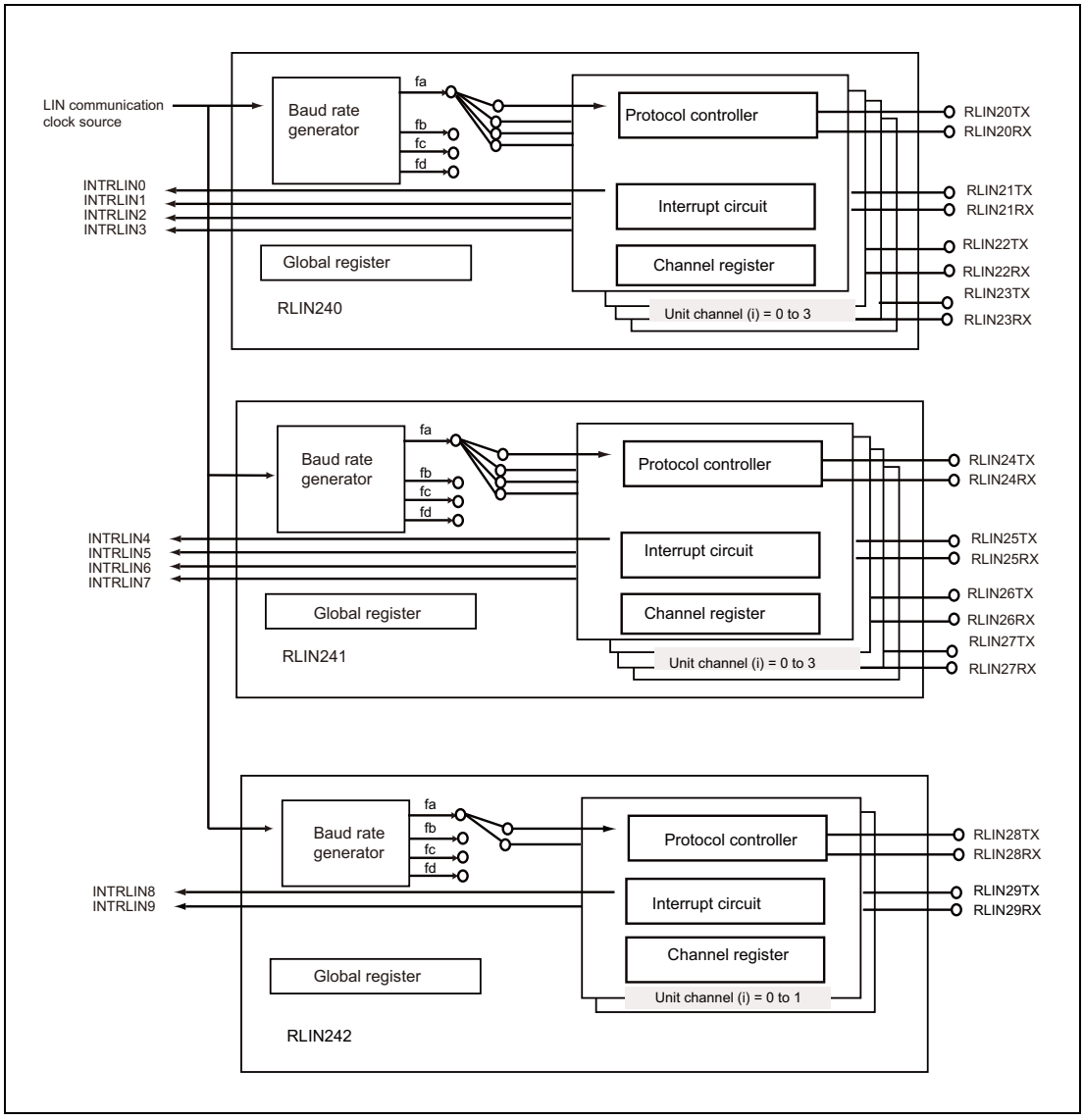



**Figure 18.1 LIN Master Interface Block Diagram (176 Pins, RLIN2 10 Channels Embedded)**

## 18.3 Registers

The registers of the LIN master interface include global registers and channel registers. As the global registers are allocated for each unit, they can be individually set for each unit. As the channel registers are allocated for each channel, they can individually control each channel.

### 18.3.1 List of Registers

RLIN2 registers are listed in the following table.

For details about <RLIN24n_base>, see **Section 18.1.2, Register Base Addresses**.

**Table 18.11 Registers**

| Module | Register | Symbol | Address |
|---|---|---|---|
| **Global registers** | | | |
| RLN24n | LIN wake-up baud rate select register | RLN24nGLWBR | <RLIN24n_base> + $01_H$ |
| | LIN baud rate prescaler 0 register | RLN24nGLBRP0 | <RLIN24n_base> + $02_H$ |
| | LIN baud rate prescaler 1 register | RLN24nGLBRP1 | <RLIN24n_base> + $03_H$ |
| | LIN self test control register | RLN24nGLSTC | <RLIN24n_base> + $04_H$ |
| **Channel registers** | | | |
| RLN24nm | LIN mode register | RLN24nmLiMD | <RLIN24n_base> + $08_H$ + i × $20_H$ |
| | LIN break field configuration register | RLN24nmLiBFC | <RLIN24n_base> + $09_H$ + i × $20_H$ |
| | LIN space configuration register | RLN24nmLiSC | <RLIN24n_base> + $0A_H$ + i × $20_H$ |
| | LIN wake-up configuration register | RLN24nmLiWUP | <RLIN24n_base> + $0B_H$ + i × $20_H$ |
| | LIN interrupt enable register | RLN24nmLiIE | <RLIN24n_base> + $0C_H$ + i × $20_H$ |
| | LIN error detection enable register | RLN24nmLiEDE | <RLIN24n_base> + $0D_H$ + i × $20_H$ |
| | LIN control register | RLN24nmLiCUC | <RLIN24n_base> + $0E_H$ + i × $20_H$ |
| | LIN transmission control register | RLN24nmLiTRC | <RLIN24n_base> + $10_H$ + i × $20_H$ |
| | LIN mode status register | RLN24nmLiMST | <RLIN24n_base> + $11_H$ + i × $20_H$ |
| | LIN status register | RLN24nmLiST | <RLIN24n_base> + $12_H$ + i × $20_H$ |
| | LIN error status register | RLN24nmLiEST | <RLIN24n_base> + $13_H$ + i × $20_H$ |
| | LIN data field configuration register | RLN24nmLiDFC | <RLIN24n_base> + $14_H$ + i × $20_H$ |
| | LIN ID buffer register | RLN24nmLiIDB | <RLIN24n_base> + $15_H$ + i × $20_H$ |
| | LIN check sum buffer register | RLN24nmLiCBR | <RLIN24n_base> + $16_H$ + i × $20_H$ |
| | LIN data buffer 1 register | RLN24nmLiDBR1 | <RLIN24n_base> + $18_H$ + i × $20_H$ |
| | LIN data buffer 2 register | RLN24nmLiDBR2 | <RLIN24n_base> + $19_H$ + i × $20_H$ |
| | LIN data buffer 3 register | RLN24nmLiDBR3 | <RLIN24n_base> + $1A_H$ + i × $20_H$ |
| | LIN data buffer 4 register | RLN24nmLiDBR4 | <RLIN24n_base> + $1B_H$ + i × $20_H$ |
| | LIN data buffer 5 register | RLN24nmLiDBR5 | <RLIN24n_base> + $1C_H$ + i × $20_H$ |
| | LIN data buffer 6 register | RLN24nmLiDBR6 | <RLIN24n_base> + $1D_H$ + i × $20_H$ |
| | LIN data buffer 7 register | RLN24nmLiDBR7 | <RLIN24n_base> + $1E_H$ + i × $20_H$ |
| | LIN data buffer 8 register | RLN24nmLiDBR8 | <RLIN24n_base> + $1F_H$ + i × $20_H$ |

**Note:** When writing to a register not used, write the value after reset.

## 18.3.2 Global Registers

### 18.3.2.1 RLN24nGLWBR — LIN Wake-Up Baud Rate Select Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nGLWBR: <RLIN24n_base> + $01_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | LWBR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 18.12 RLN24nGLWBR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | LWBR0 | Wake-up Baud Rate Select<br>0: In LIN wake-up mode, the clock specified by the LCKS bit setting in the RLN24nmLiMD registers is used (when LIN 1.3 is used).<br>1: In LIN wake-up mode, the clock fa is used regardless of the setting of the LCKS bit in the RLN24nmLiMD registers (when LIN 2.X is used). |

Set the RLN24nGLWBR register when all channels in the same unit are in LIN reset mode (the OMM0 bit in the RLN24nmLiMST register is 0).

#### LWBR0 Bit (Wake-Up Baud Rate Select)

When LIN Specification Package Revision 1.3 is used, set the LWBR0 bit in the RLN24nGLWBR register to 0. This allows the 2.5-Tbit or longer low-level width of the input signal to be measured.
When LIN Specification Package Revision 2.x is used, set the LWBR0 bit to 1. With this setting, fa is selected as the LIN system clock ($f_{LIN}$) during LIN wake-up mode regardless of the setting of the LCKS bit in the RLN24nmLiMD register (the LCKS bit is not changed) and the 2.5-Tbit or longer low-level width of the input signal can be measured.
Setting the baud rate to 19200 bps while fa is selected allows the 130 µs or longer low-level width of the input signal to be detected during LIN wake-up mode regardless of the setting of the LCKS bit in the RLN24nmLiMD register.

### 18.3.2.2 RLN24nGLBRP0 — LIN Baud Rate Prescaler 0 Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nGLBRP0: <RLIN24n_base> + $02_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LBRP0[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.13 RLN24nGLBRP0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | LBRP0[7:0] | Assuming that the value set in this register is N (0 to 255), the baud rate prescaler divides the LIN communication clock source by N+1.<br>Setting range: $00_H$ to $FF_H$ |

Set the RLN24nGLBRP0 register when all channels in the same unit are in LIN reset mode (the OMM0 bit in the RLN24nmLiMST register is 0).

The value set in this register is used to control the frequency of baud rate clock source fa, fb, and fc.

Assuming that the value set in this register is N, baud rate prescaler 0 divides the LIN communication clock source by N+1.

### 18.3.2.3 RLN24nGLBRP1 — LIN Baud Rate Prescaler 1 Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nGLBRP1: <RLIN24n_base> + $03_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LBRP1[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.14 RLN24nGLBRP1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | LBRP1[7:0] | Assuming that the value set in this register is M (0 to 255), the baud rate prescaler divides the LIN communication clock source by M+1.<br>Setting range: $00_H$ to $FF_H$ |

Set the RLN24nGLBRP1 register when all channels in the same unit are in LIN reset mode (the OMM0 bit in the RLN24nmLiMST register is 0).

The value set in this register is used to control the frequency of baud rate clock source fd.

Assuming that the value set in this register is M, baud rate prescaler 1 divides the LIN communication clock source by M+1.

### 18.3.2.4 RLN24nGLSTC — LIN Self-Test Control Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nGLSTC: <RLIN24n_base> + $04_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | LSTM |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.15 RLN24nGLSTC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | — | Writing $A7_H$, $58_H$, and $01_H$ successively to the RLN24nGLSTC register places the module in LIN self-test mode. |
| 0 | LSTM | LIN Self-Test Mode<br>0: The module is not in LIN self-test mode.<br>1: The module is in LIN self-test mode. |

The RLN24nGLSTC register cancels protection of LIN self-test mode.

Set the RLN24nGLSTC register when all channels in the same unit are in LIN reset mode (the OMM0 bit in the RLN24nmLiMST register is 0).

Writing $A7_H$, $58_H$, and $01_H$ successively to the RLN24nGLSTC register places the module into LIN self-test mode.

When successive writing is completed and the module is placed in LIN self-test mode, the LSTM bit is set to 1.

Do not write any other value during successive writing.

For details about transitioning to LIN self-test mode, see **Section 18.15, LIN Self-Test Mode**.

When read, bits 6 to 1 return "$000000_B$", and bit 7 returns an undefined value.

#### LSTM Bit (LIN Self-Test Mode)

When transition to LIN self-test mode is completed, the LSTM bit is set to 1.
For details about exiting LIN self-test mode, see **Section 18.15, LIN Self-Test Mode**.
Writing 1 to this bit does not affect the value of the RLN24nGLSTC register if it is not a part of successive writing of $A7_H$, $58_H$, and $01_H$.

## 18.3.3 Channel Registers

### 18.3.3.1 RLN24nmLiMD — LIN Mode Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nmLiMD: <RLIN24n_base> + $08_H$ + i × $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | LCKS[1:0] | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R | R |

**Table 18.16 RLN24nmLiMD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3, 2 | LCKS[1:0] | LIN System Clock Select<br>b3 b2<br>0 0: fa (Clock generated by baud rate prescaler 0)<br>0 1: fb (1/2 clock generated by baud rate prescaler 0)<br>1 0: fc (1/8 clock generated by baud rate prescaler 0)<br>1 1: fd (1/2 clock generated by baud rate prescaler 1) |
| 1, 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Set the RLN24nmLiMD register while the OMM0 bit in the RLN24nmLiMST register is 0 (LIN reset mode).

#### LCKS[1:0] Bits (LIN System Clock Select)

The LCKS bits select the clock to be input to the protocol controller.
With $00_B$ set, the protocol controller is provided with fa (clock generated by baud rate prescaler 0).
With $01_B$ set, the protocol controller is provided with fb (1/2 clock generated by baud rate prescaler 0).
With $10_B$ set, the protocol controller is provided with fc (1/8 clock generated by baud rate prescaler 0).
With $11_B$ set, the protocol controller is provided with fd (1/2 clock generated by baud rate prescaler 1).

When the LWBR0 bit in the RLN24nGLWBR is 1 (when LIN 2.x is used) and the RLN24nmLiMST register is $01_H$ (LIN wake-up mode), regardless of the setting of the LCKS bit, fa is input to the protocol controller (LCKS bit is not changed).

### 18.3.3.2 RLN24nmLiBFC — LIN Break Field Configuration Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nmLiBFC: <RLIN24n_base> + $09_H$ + i × $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | BDT[1:0] | | BLT[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.17 RLN24nmLiBFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5, 4 | BDT[1:0] | Transmission Break Delimiter (High Level) Width Select<br>b5 b4<br>0 0: 1 Tbit<br>0 1: 2 Tbits<br>1 0: 3 Tbits<br>1 1: 4 Tbits |
| 3 to 0 | BLT[3:0] | Transmission Break (Low Level) Width Select<br>b3 b0<br>0 0 0 0: 13 Tbits<br>0 0 0 1: 14 Tbits<br>0 0 1 0: 15 Tbits<br>:<br>1 1 1 0: 27 Tbits<br>1 1 1 1: 28 Tbits |

Set the RLN24nmLiBFC register while the OMM0 bit in the RLN24nmLiMST register is 0 (LIN reset mode).

Some combinations of the set values result in the length of a frame exceeding the timeout time. Set the appropriate values in this register.

#### BDT[1:0] Bits (Transmission Break Delimiter (High Level) Width Select)

The BDT bits set the break delimiter (high level) width of transmission frame header.
1 Tbit to 4 Tbits can be specified.

#### BLT[3:0] Bits (Transmission Break (Low Level) Width Select)

The BLT bits set the break (low level) width of transmission frame header.
13 Tbits to 28 Tbits can be specified.

### 18.3.3.3 RLN24nmLiSC — LIN Space Configuration Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nmLiSC: <RLIN24n_base> + $0A_H$ + i × $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | IBS[1:0] | | — | IBHS[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R | R/W | R/W | R/W |

**Table 18.18 RLN24nmLiSC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5, 4 | IBS[1:0] | Inter-Byte Space Select<br>b5 b4<br>0 0: 0 Tbit<br>0 1: 1 Tbit<br>1 0: 2 Tbits<br>1 1: 3 Tbits |
| 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 to 0 | IBHS[2:0] | Inter-Byte Space (Header)/Response Space Select<br>b2 b0<br>0 0 0: 0 Tbit<br>0 0 1: 1 Tbit<br>0 1 0: 2 Tbits<br>0 1 1: 3 Tbits<br>1 0 0: 4 Tbits<br>1 0 1: 5 Tbits<br>1 1 0: 6 Tbits<br>1 1 1: 7 Tbits |

Set the RLN24nmLiSC register while the OMM0 bit in the RLN24nmLiMST register is 0 (LIN reset mode).

Some combinations of the set values result in the length of a frame or a response exceeding the timeout time. Set the appropriate values in this register.

#### IBS[1:0] Bits (Inter-Byte Space Select)

The IBS bits set the width of the inter-byte space of the transmission frame response field.
0 Tbit to 3 Tbits can be specified.
These bits are enabled only during response transmission; they are disabled during response reception.

#### IBHS[2:0] Bits (Inter-Byte Space (Header)/Response Space Select)

The IBHS bits set the width of the inter-byte space (header) of the transmission frame header field and the response space.
0 Tbit to 7 Tbits can be specified.
The response space setting is enabled only during response transmission; the setting is disabled during response reception.
The inter-byte space (header) value is equal to the response space value.

### 18.3.3.4 RLN24nmLiWUP — LIN Wake-Up Configuration Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nmLiWUP: <RLIN24n_base> + $0B_H$ + i × $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WUTL[3:0] | | | | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R | R | R |

**Table 18.19 RLN24nmLiWUP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | WUTL[3:0] | Wake-up Transmission Low Level Width Select<br>b7 b4<br>0 0 0 0: 1 Tbit<br>0 0 0 1: 2 Tbits<br>0 0 1 0: 3 Tbits<br>0 0 1 1: 4 Tbits<br>:<br>1 1 0 0: 13 Tbits<br>1 1 0 1: 14 Tbits<br>1 1 1 0: 15 Tbits<br>1 1 1 1: 16 Tbits |
| 3 to 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Set the RLN24nmLiWUP register while the OMM0 bit in the RLN24nmLiMST register is 0 (LIN reset mode).

#### WUTL[3:0] Bits (Wake-Up Transmission Low Level Width Select)

The WUTL bits set the low level width of the wake-up signal transmission.
1 Tbit to 16 Tbits can be specified.
When the LWBR0 bit in the RLN24nGLWBR is 1 (when LIN 2.x is used), regardless of the setting of the LCKS bit in the RLN24nmLiMD register, fa is selected as the LIN system clock ($f_{LIN}$) (LCKS bit is not changed).

### 18.3.3.5 RLN24nmLiIE — LIN Interrupt Enable Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nmLiIE: <RLIN24n_base> + $0C_H$ + i × $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | ERRIE | FRCIE | FTCIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W |

**Table 18.20 RLN24nmLiIE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 | ERRIE | Error Detection Interrupt Request Enable<br>0: Disables error detection interrupt request.<br>1: Enables error detection interrupt request. |
| 1 | FRCIE | Successful Frame/Wake-up Reception Interrupt Request Enable<br>0: Disables successful frame/wake-up reception interrupt request.<br>1: Enables successful frame/wake-up reception interrupt request. |
| 0 | FTCIE | Successful Frame/Wake-up Transmission Interrupt Request Enable<br>0: Disables successful frame/wake-up transmission interrupt request.<br>1: Enables successful frame/wake-up transmission interrupt request. |

Set the RLN24nmLiIE register while the OMM0 bit in the RLN24nmLiMST register is 0 (LIN reset mode).

#### ERRIE Bit (Error Detection Interrupt Request Enable)

The ERRIE bit enables or disables interrupt request upon detection of an error.
With 0 set, the interrupt request is not generated when the ERR flag in the RLN24nmLiST register is set to 1.
With 1 set, the interrupt request is generated when the ERR flag in the RLN24nmLiST register is set to 1.
Errors that constitute interrupt sources are bit errors, physical bus errors, frame timeout errors, framing errors, and checksum errors.
Detection of a bit error, physical bus error, frame timeout error, and framing error can be enabled or disabled using the RLN24nmLiEDE register.

#### FRCIE Bit (Successful Frame/Wake-Up Reception Interrupt Request Enable)

The FRCIE bit enables or disables interrupt request upon successful reception of a frame or a wake-up signal (counting of low level width of the input signal).
With 0 set, the interrupt request is not generated when the FRC flag in the RLN24nmLiST register is set to 1.
With 1 set, the interrupt request is generated when the FRC flag in the RLN24nmLiST register is set to 1.

#### FTCIE Bit (Successful Frame/Wake-Up Transmission Interrupt Request Enable)

The FTCIE bit enables or disables interrupt request upon successful transmission of a frame or a wake-up signal.
With 0 set, the interrupt request is not generated when the FTC flag in the RLN24nmLiST register is set to 1.
With 1 set, the interrupt request is generated when the FTC flag in the RLN24nmLiST register is set to 1.

### 18.3.3.6 RLN24nmLiEDE — LIN Error Detection Enable Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nmLiEDE: <RLIN24n_base> + $0D_H$ + i × $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | FERE | FTERE | PBERE | BERE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 18.21 RLN24nmLiEDE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | FERE | Framing Error Detection Enable<br>0: Disables framing error detection.<br>1: Enables framing error detection. |
| 2 | FTERE | Frame Timeout Error Detection Enable<br>0: Disables frame timeout error detection.<br>1: Enables frame timeout error detection. |
| 1 | PBERE | Physical Bus Error Detection Enable<br>0: Disables physical bus error detection.<br>1: Enables physical bus error detection. |
| 0 | BERE | Bit Error Detection Enable<br>0: Disables bit error detection.<br>1: Enables bit error detection. |

Set the RLN24nmLiEDE register while the OMM0 bit in the RLN24nmLiMST register is $0_B$ (LIN reset mode).

#### FERE Bit (Framing Error Detection Enable)

The FERE bit enables or disables detection of the framing error.
With 0 set, the framing error is not detected.
With 1 set, the framing error is detected.
When this bit is set to 1, the detection result is indicated in the FER flag in the RLN24nmLiEST register.
For details on the framing error, see **Section 18.14, Error Status**.

#### FTERE Bit (Frame Timeout Error Detection Enable)

The FTERE bit enables or disables detection of the frame timeout error.
With 0 set, the frame timeout error is not detected.
With 1 set, the frame timeout error is detected.
When this bit is set to 1, the detection result is indicated in the FTER flag in the RLN24nmLiEST register.
For details on the frame timeout error, see **Section 18.14, Error Status**.

#### PBERE Bit (Physical Bus Error Detection Enable)

The PBERE bit enables or disables detection of the physical bus error.
With 0 set, the physical bus error is not detected.
With 1 set, the physical bus error is detected.
When this bit is set to 1, the detection result is indicated in the PBER flag in the RLN24nmLiEST register.
For details on the physical bus error, see **Section 18.14, Error Status**.

#### BERE Bit (Bit Error Detection Enable)

The BERE bit enables or disables detection of the bit error.
With 0 set, the bit error is not detected.
With 1 set, the bit error is detected.
When this bit is set to 1, the detection result is indicated in the BER flag in the RLN24nmLiEST register.
For details on the bit error, see **Section 18.14, Error Status**.

### 18.3.3.7 RLN24nmLiCUC — LIN Control Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nmLiCUC: <RLIN24n_base> + $0E_H$ + i × $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | OM1 | OM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 18.22 RLN24nmLiCUC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 1 | OM1 | LIN Mode Select<br>0: LIN wake-up mode.<br>1: LIN operation mode. |
| 0 | OM0 | LIN Reset<br>0: Transition to LIN reset mode.<br>1: Exit LIN reset mode. |

Set the RLN24nmLiCUC register to $01_H$ to transition to LIN wake-up mode or to $03_H$ to transition to LIN operation mode after exiting LIN reset mode.

In LIN self-test mode, set the RLN24nmLiCUC register to $03_H$ after a transition to LIN self-test mode is completed.

After a value is written to this register, confirm that the value written is actually indicated in the RLN24nmLiMST register before writing another value.

#### OM1 Bit (LIN Mode Select)

The OM1 bit selects the operating mode (LIN wake-up mode or LIN operation mode) that is entered after exiting LIN reset mode.
Setting this bit to 0 selects LIN wake-up mode.
Setting this bit to 1 selects LIN operation mode.
This bit is enabled only when the OMM0 bit in the RLN24nmLiMST register is 1.
Writing a value to this bit is disabled while the FTS bit in the RLN24nmLiTRC register is 1.

#### OM0 Bit (LIN Reset)

The OM0 bit selects whether to transition to or exit LIN reset mode.
Setting this bit to 0 causes RLIN2 to enter LIN reset mode.
Setting this bit to 1 causes RLIN2 to exit LIN reset mode.

### 18.3.3.8 RLN24nmLiTRC — LIN Transmission Control Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nmLiTRC: <RLIN24n_base> + $10_H$ + i × $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | RTS | FTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 18.23 RLN24nmLiTRC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 1 | RTS | Response Transmission Start<br>0: Response transmission is stopped in frame separate mode.<br>1: Response transmission is started in frame separate mode. |
| 0 | FTS | Frame Transmission/Wake-up Transmission/Reception Start<br>0: Frame Transmission/wake-up transmission/reception is stopped.<br>1: Frame Transmission/wake-up transmission/reception is started. |

#### RTS Bit (Response Transmission Start)

Set the RTS bit (response transmit start bit) to 1 in frame separate mode after header transmission is started (FTS bit is 1) and response transmission data is ready. Once set, this bit is automatically cleared to 0 upon completion of frame transmission and transition to LIN reset mode.
Only 1 can be written to this bit; 0 cannot be written.
To write 1 to this bit, write $02_H$ to the RLN24nmLiTRC register using the store instruction.
Writing a value to this bit is disabled when the OMM0 bit is 0 (LIN reset mode). When the OMM1 bit is 0 (LIN wake-up mode), do not write 1.
Writing a value to this bit is disabled when the FTS bit is 0 (frame transmission or wake-up transmission/reception is stopped).

#### FTS Bit (Frame Transmission/Wake-Up Transmission/Reception Start)

Set the FTS bit to 1 to start frame/wake-up transmission.
Also set this bit to 1 to allow wake-up reception (counting of the low level width of the input signal).
Only 1 can be written to this bit; 0 cannot be written.
Writing a value to this bit is disabled when the OMM0 bit is 0 (LIN reset mode).
This bit is set to 0 upon completion of frame or wake-up communication and transition to LIN reset mode.

### 18.3.3.9 RLN24nmLiMST — LIN Mode Status Register

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** RLN24nmLiMST: <RLIN24n_base> + $11_H$ + i × $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | OMM1 | OMM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 18.24 RLN24nmLiMST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. |
| 1 | OMM1 | LIN Mode Status Monitor<br>0: LIN wake-up mode.<br>1: LIN operation mode. |
| 0 | OMM0 | LIN Reset Status Monitor<br>0: LIN reset mode.<br>1: Not in LIN reset mode. |

#### OMM1 Bit (LIN Mode Status Monitor)

The OMM1 bit indicates the current operating mode.
When the OMM0 bit is $0_B$ (LIN reset mode), the value of this bit is disabled.

#### OMM0 Bit (LIN Reset Status Monitor)

The OMM0 bit indicates the current operating mode.

### 18.3.3.10 RLN24nmLiST — LIN Status Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nmLiST: <RLIN24n_base> + $12_H$ + i × $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | HTRC | D1RC | — | — | ERR | — | FRC | FTC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R/W | R/W |

**Table 18.25 RLN24nmLiST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | HTRC | Successful Header Transmission Flag<br>0: Header transmission has not been completed.<br>1: Header transmission has been completed. |
| 6 | D1RC | Successful Data 1 Reception Flag<br>0: Data 1 reception has not been completed.<br>1: Data 1 reception has been completed. |
| 5, 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | ERR | Error Detection Flag<br>0: No error has been detected.<br>1: Error has been detected. |
| 2 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 1 | FRC | Successful Frame/Wake-up Reception Flag<br>0: Frame or wake-up reception has not been completed.<br>1: Frame or wake-up reception has been completed. |
| 0 | FTC | Successful Frame/Wake-up Transmission Flag<br>0: Frame or wake-up transmission has not been completed.<br>1: Frame or wake-up transmission has been completed. |

The RLN24nmLiST register is automatically cleared to $00_H$ upon transition to LIN reset mode and start of the next communication is started (when the value of the FTS bit of the RLN24nmLiTRC register is 1).

In LIN reset mode, this register cannot be written to. In LIN reset mode, the register retains $00_H$.

Writing to this register is prohibited while the FTS bit in the RLN24nmLiTRC register is 1 (frame transmission or wake-up transmission/reception is started)

To clear the specific bits in the register, write 0 to the bits to be cleared and write 1 to the other bits using the store instruction.

#### HTRC Flag (Successful Header Transmission Flag)

Only 0 can be written to the HTRC flag; when 1 is written, the bit retains the value from before 1 was written.
"1" is set upon completion of header transmission, but an interrupt request is not generated.
To clear the bit to 0 before the next communication is started (when the value of the FTS bit of the RLN24nmLiTRC register is 1), write 0 to the bit while in LIN operation mode.

#### D1RC Flag (Successful Data 1 Reception Flag)

Only 0 can be written to the D1RC flag; when 1 is written, the bit retains the value from before 1 is written.
"1" is set upon completion of Data 1 reception, but an interrupt request is not generated.
To clear the bit to 0 before the next communication is started (when the value of the FTS bit of the RLN24nmLiTRC register is 1), write 0 to the bit while in LIN operation mode.

#### ERR Flag (Error Detection Flag)

The ERR flag is set to 1 upon detection of an error (when the value of any of the flags of the RLN24nmLiEST registers is 1). Here, an interrupt request is generated if the ERRIE bit in the RLN24nmLiIE register is 1 (interrupt is enabled). To clear the bit to 0 before the next communication is started (when the value of the FTS bit of the RLN24nmLiTRC register is 1), write 0 to the CSER flag, FER flag, FTER flag, PBER flag, and BER flag in the RLN24nmLiEST register while in LIN operation mode or LIN wake-up mode. This clears the ERR flag to 0.

#### FRC Flag (Successful Frame/Wake-Up Reception Flag)

Only 0 can be written to the FRC flag; when 1 is written, the bit retains the value from before 1 is written.
The FRC flag is set to 1 upon completion of frame or wake-up reception. Here, an interrupt request is generated if the FRCIE bit in the RLN24nmLiIE register is 1 (interrupt is enabled).
To clear the bit to 0 before the next communication is started (when the value of the FTS bit of the RLN24nmLiTRC register is 1), write 0 to the bit while in LIN operation mode or LIN wake-up mode.

#### FTC Flag (Successful Frame/Wake-Up Transmission Flag)

Only 0 can be written to the FTC flag; when 1 is written, the bit retains the value from before 1 is written.
The FTC flag is set to 1 upon completion of frame or wake-up transmission. Here, an interrupt request is generated if the FTCIE bit in the RLN24nmLiIE register is 1 (interrupt is enabled).
To clear the bit to 0 before the next communication is started (when the value of the FTS bit of the RLN24nmLiTRC register is 1), write 0 to the bit while in LIN operation mode or LIN wake-up mode.

### 18.3.3.11 RLN24nmLiEST — LIN Error Status Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nmLiEST: <RLIN24n_base> + $13_H$ + i × $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | CSER | — | FER | FTER | PBER | BER |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R | R/W | R/W | R/W | R/W |

**Table 18.26 RLN24nmLiEST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5 | CSER | Checksum Error Flag<br>0: Checksum error has not been detected.<br>1: Checksum error has been detected. |
| 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | FER | Framing Error Flag<br>0: Framing error has not been detected.<br>1: Framing error has been detected. |
| 2 | FTER | Frame Timeout Error Flag<br>0: Frame timeout error has not been detected.<br>1: Frame timeout error has been detected. |
| 1 | PBER | Physical Bus Error Flag<br>0: Physical bus error has not been detected.<br>1: Physical bus error has been detected. |
| 0 | BER | Bit Error Flag<br>0: Bit error has not been detected.<br>1: Bit error has been detected. |

The RLN24nmLiEST register is automatically cleared to $00_H$ upon transition to LIN reset mode and start of the next communication is started (when the value of the FTS bit of the RLN24nmLiTRC register is 1).

In LIN reset mode, this register cannot be written to. In LIN reset mode, the register retains $00_H$.

When the FTS bit in the RLN24nmLiTRC register is 1 (frame transmission or wake-up transmission/ reception is started), do not write a value to this register.

To clear the specific bits in the register, write 0 to the bits to be cleared and write 1 to the other bits using the store instruction.

#### CSER Flag (Checksum Error Flag)

Only 0 can be written to the CSER flag; when 1 is written, the bit retains the value from before 1 is written.
The CSER flag is set to 1 upon checksum error detection. To clear the bit to 0 before the next communication is started (when the value of the FTS bit of the RLN24nmLiTRC register is 1), write 0 to the bit while in LIN operation mode.

### FER Flag (Framing Error Flag)

Only 0 can be written to the FER flag; when 1 is written, the bit retains the value from before 1 is written.
The FER flag is set to 1 upon framing error detection when the FERE bit of the RLN24nmLiEST register is 1 (framing error detection enabled). To clear the bit to 0 before the next communication is started (when the value of the FTS bit of the RLN24nmLiTRC register is 1), write 0 to the bit while in LIN operation mode.

### FTER Flag (Frame Timeout Error Flag)

Only 0 can be written to the FTER flag; when 1 is written, the bit retains the value from before 1 is written.
The FTER flag is set to 1 upon frame timeout error detection when the FTERE bit of the RLN24nmLiEDE register is 1 (frame timeout error detection enabled). To clear the bit to 0 before the next communication is started (when the value of the FTS bit of the RLN24nmLiTRC register is 1), write 0 to the bit while in LIN operation mode.

### PBER Flag (Physical Bus Error Flag)

Only 0 can be written to the PBER flag; when 1 is written, the bit retains the value from before 1 is written.
The PBER flag is set to 1 upon physical bus error detection when the PBERE bit of the RLN24nmLiEDE register is 1 (physical bus error detection enabled). To clear the bit to 0 before the next communication is started (when the value of the FTS bit of the RLN24nmLiTRC register is 1), write 0 to the bit while in LIN operation mode or LIN wake-up mode.

### BER Flag (Bit Error Flag)

Only 0 can be written to the BER flag; when 1 is written, the bit retains the value from before 1 is written.
The BER flag is set to 1 upon bit error detection when the BERE bit of the RLN24nmLiEDE register is 1 (bit error detection enabled). To clear the bit to 0 before the next communication is started (when the value of the FTS bit of the RLN24nmLiTRC register is 1), write 0 to the bit while in LIN operation mode or LIN wake-up mode.

### 18.3.3.12 RLN24nmLiDFC — LIN Data Field Configuration Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nmLiDFC: <RLIN24n_base> + $14_H$ + i × $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | FSM | CSM | RFT | RFDL[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.27 RLN24nmLiDFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 6 | FSM | Frame Separate Mode Select<br>0: Frame separate mode is not set.<br>1: Frame separate mode is set. |
| 5 | CSM | Checksum Select<br>0: Classic checksum mode<br>1: Enhanced checksum mode |
| 4 | RFT | Response Field Communication Direction Select<br>0: Reception<br>1: Transmission |
| 3 to 0 | RFDL[3:0] | Response Field Length Select<br>b3 b0<br>0 0 0 0: 0 bytes + checksum<br>0 0 0 1: 1 byte + checksum<br>0 0 1 0: 2 bytes + checksum<br>:<br>0 1 1 1: 7 bytes + checksum<br>1 0 0 0: 8 bytes + checksum<br>Settings other than the above are prohibited. |

Set the RLN24nmLiDFC register when the FTS bit in the RLN24nmLiTRC register is 0 (frame transmission or wake-up transmission/reception is stopped).

#### FSM Bit (Frame Separate Mode Select)

The FSM bit sets the response transmission mode.

With 0 set, frame separate mode is not selected. In this case, after header transmission is started (the FTS bit in the RLN24nmLiTRC register is 1), response is transmitted/received without the RTS bit in the RLN24nmLiTRC register being set.
With 1 set, frame separate mode is selected. When the RTS bit of the RLN24nmLiTRC register is set to 1 during header transmission, response transmission is executed after header transmission has ended.
For response reception (the RFT bit is 0), set the FSM bit to 0.
When transitioning to LIN self-test mode, set this bit to 0 before transition.
For details on frame separate mode, see **Section 18.11.1, Transmission of LIN Frames**.

#### CSM Bit (Checksum Select)

The CSM bit sets checksum mode.
With 0 set, classic checksum mode is selected.
With 1 set, enhanced checksum mode is selected.
When frame timeout error detection is enabled (the FTERE bit in the RLN24nmLiEDE register is 1), the timeout time depends on the setting of this bit. For details, see **Section 18.14, Error Status**.

#### RFT Bit (Response Field Communication Direction Select)

The RFT bit sets the direction of the response field/wake-up signal communication.
With 0 set, reception is performed in the response field. In LIN wake-up mode, wake-up reception is performed (low level width of the input signal is counted).
With 1 set, transmission is performed in the response field. In LIN wake-up mode, wake-up transmission is performed.

#### RFDL[3:0] Bits (Response Field Length Select)

The RFDL bits set the length of the response field data.
The data length can be 0 to 8 bytes excluding the checksum size.

### 18.3.3.13 RLN24nmLiIDB — LIN ID Buffer Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nmLiIDB: <RLIN24n_base> + $15_H$ + i × $20_H$

**Value after reset:** Undefined

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IDP1 | IDP0 | ID[5:0] | | | | | |
| Value after reset | — | — | — | — | — | — | — | — |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.28 RLN24nmLiIDB Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | IDP1 | Parity Setting (P1)<br>Sets the parity bit (P1) to be transmitted in the ID field. |
| 6 | IDP0 | Parity Setting (P0)<br>Sets the parity bit (P0) to be transmitted in the ID field. |
| 5 to 0 | ID[5:0] | ID Setting<br>Sets the 6-bit ID value to be transmitted in the ID field. |

Set the RLN24nmLiIDB register when the FTS bit in the RLN24nmLiTRC register is 0 (frame transmission or wake-up transmission/reception is stopped).

In LIN self-test mode, the operation is as follows.

Write the value to be transmitted prior to communication. After completion of frame transmission/ reception (after loopback), the reversed value of the received value can be read.

For details about LIN self-test mode, see **Section 18.15, LIN Self-Test Mode**.

#### IDP[1:0] Bits (Parity Setting)

The IDP bits set the parity bits (P0 and P1) to be transmitted in the ID field of the LIN frame. IDP0 sets P0 and IDP1 sets P1.
Since parity is not automatically calculated, set the calculation value. Note that even if the specified calculation result is incorrect, it is transmitted as is.

#### ID[5:0] Bits (ID Setting)

The ID bits set the 6-bit ID value to be transmitted in the ID field of the LIN frame.

### 18.3.3.14 RLN24nmLiCBR — LIN Checksum Buffer Register

**Access:** This register is a read-only register that can be read in 8-bit units. In LIN self-test mode, this register can be read or written in 8-bit units.

**Address:** RLN24nmLiCBR: <RLIN24n_base> + $16_H$ + i × $20_H$

**Value after reset:** Undefined

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CKSM[7:0] | | | | | | | |
| Value after reset | — | — | — | — | — | — | — | — |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.29 RLN24nmLiCBR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | CKSM[7:0] | Holds the checksum value transmitted or received. |

while in LIN operation mode, this register operates as follows:

- When the RFT bit in the RLN24nmLiDFC register is 1 (transmission):
  The value transmitted can be read from the register. Read the value after transmission is completed.
  Writing to this register is invalid.
- When the RFT bit in the RLN24nmLiDFC register is 0 (reception):
  The value received can be read from the register. Read the value after reception is completed.
  Writing to this register is invalid.

In LIN self-test mode, this register operates as follows:

- When the RFT bit in the RLN24nmLiDFC register is 1 (transmission):
  After completion of frame transmission/reception (after loopback), the reversed value of the received value can be read.
  Writing to this register is invalid.
- When the RFT bit in the RLN24nmLiDFC register is 0 (reception):
  Write the value to be received before communication. After completion of frame transmission/reception (after loopback), the reversed value of the received value can be read.

For details about LIN self-test mode, see **Section 18.15, LIN Self-Test Mode**.

Set the RLN24nmLiCBR register when the FTS bit in the RLN24nmLiTRC register is 0 (frame transmission or wake-up transmission/reception is stopped).

### 18.3.3.15 RLN24nmLiDBRb — LIN Data Buffer b Register

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN24nmLiDBR1: <RLIN24n_base> + $18_H$ + i × $20_H$
RLN24nmLiDBR2: <RLIN24n_base> + $19_H$ + i × $20_H$
RLN24nmLiDBR3: <RLIN24n_base> + $1A_H$ + i × $20_H$
RLN24nmLiDBR4: <RLIN24n_base> + $1B_H$ + i × $20_H$
RLN24nmLiDBR5: <RLIN24n_base> + $1C_H$ + i × $20_H$
RLN24nmLiDBR6: <RLIN24n_base> + $1D_H$ + i × $20_H$
RLN24nmLiDBR7: <RLIN24n_base> + $1E_H$ + i × $20_H$
RLN24nmLiDBR8: <RLIN24n_base> + $1F_H$ + i × $20_H$

**Value after reset:** Undefined

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LDB[7:0] | | | | | | | |
| Value after reset | — | — | — | — | — | — | — | — |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.30 RLN24nmLiDBRb Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | LDB[7:0] | Sets the data to be transmitted or allows the received data to be read.<br>Setting range: $00_H$ to $FF_H$ |

- For response transmission:

  The LiDBRb registers specify the data to be transmitted in the response field.
  Configure these registers when:

  – The RFT bit in RLN24nmLiDFC register is 1 (transmission).
  – The FSM bit in RLN24nmLiDFC register is 0 (not frame separate mode).
  – The FTS bit in RLN24nmLiTRC register is 0 (frame transmission or wake-up transmission/ reception is stopped).

  or

  – The RFT bit in RLN24nmLiDFC register is 1 (transmission).
  – The FSM bit in RLN24nmLiDFC register is 1 (frame separate mode).
  – The RTS bit in RLN24nmLiTRC register is 0 (response transmission is stopped).

- For response reception:

  The LiDBRb registers hold the data received in the response field.
  The received data is overwritten. If an error is detected, the data prior to reception interruption is stored in the register.
  Do not read these registers when the FTS bit is 1 (frame transmission or wake-up transmission/ reception is started)

In LIN self-test mode, the operation is as follows.

Write the value to be transmitted prior to communication. After completion of frame transmission/ reception (after loopback), the reversed value of the received value can be read.

For details about LIN self-test mode, see **Section 18.15, LIN Self-Test Mode**.

## 18.4 Interrupt Sources

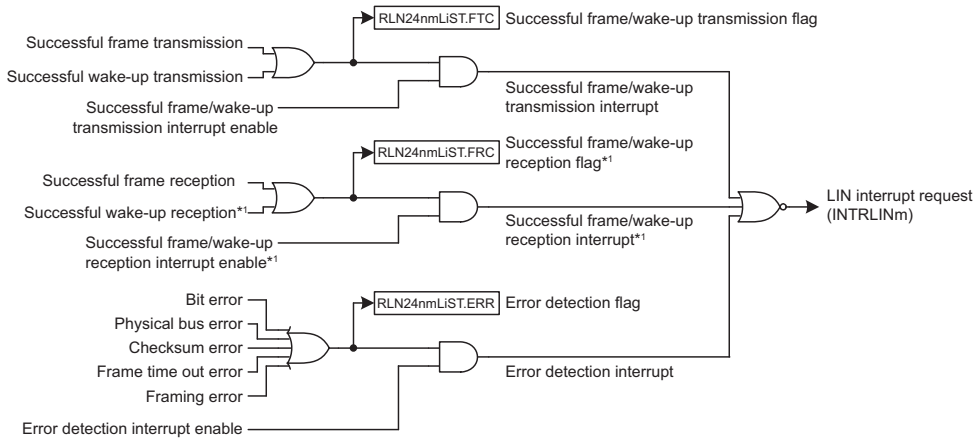
The LIN interrupts are interrupt requests generated by the LIN master interface.

There are three interrupt sources for each channel; successful frame/wake-up transmission, successful frame/wakeup reception, and error detection.

Interrupt requests from these three sources are ORed to generate one interrupt request “LIN interrupt”.

The respective interrupt request is output when the corresponding flag in the RLN24nmLiST register is set to 1 while the corresponding bit in the RLN24nmLiIE register is 1 (interrupt enabled). However, if an interrupt is requested when the corresponding flag in the RLN24nmLiST register has been set to 1, it is ignored. Therefore, clear the corresponding flag to 0 to enable the interrupt again.

**Figure 18.2** shows a block diagram of the LIN interrupt.




Note 1. The wake-up receive signal indicates the input signal low level width count.

**Figure 18.2 LIN Interrupt Block Diagram**

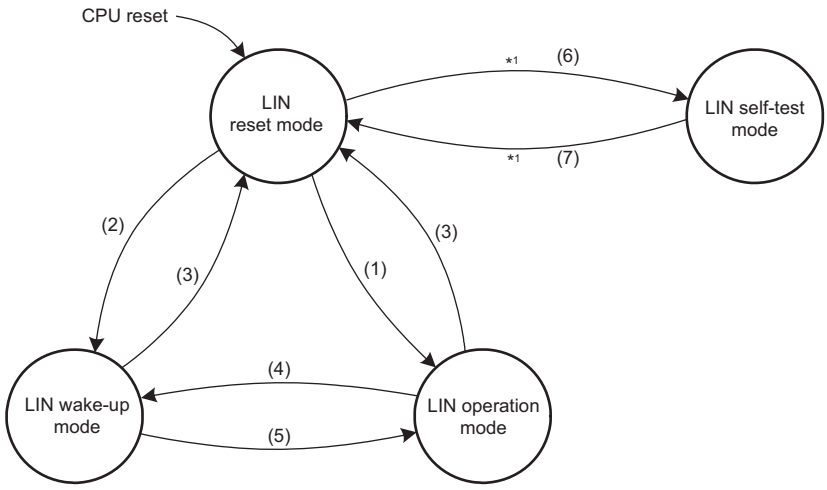
## 18.5 Modes

The LIN master interface provides the following four modes:

- LIN reset mode
- LIN operation mode
- LIN wake-up mode
- LIN self-test mode

The mode transitions except LIN self-test mode is controlled independently for respective channels.

**Figure 18.3** shows mode transitions. **Table 18.31** describes mode transition conditions. **Table 18.32** lists operations available in each mode.




Note 1. See **Section 18.15, LIN Self-Test Mode** for the details.

**Figure 18.3 Mode Transitions**

**Table 18.31 Transition Condition of Each Mode**

| | Mode Transition | | | Transition Condition |
|---|---|---|---|---|
| (1) | LIN reset mode | → | LIN operation mode | RLN24nmLiCUC.OM1,OM0 = $11_B$ |
| (2) | LIN reset mode | → | LIN wake-up mode | RLN24nmLiCUC.OM1,OM0 = $01_B$ |
| (3) | LIN wake-up mode<br>LIN operation mode | → | LIN reset mode | RLN24nmLiCUC.OM0 = $0_B$ |
| (4) | LIN operation mode | → | LIN wake-up mode | RLN24nmLiCUC.OM1,OM0 = $01_B$ |
| (5) | LIN wake-up mode | → | LIN operation mode | RLN24nmLiCUC.OM1,OM0 = $11_B$ |
| (6) | LIN reset mode | → | LIN self-test mode | See **Section 18.15, LIN Self-Test Mode**. |
| (7) | LIN self-test mode | → | LIN reset mode | See **Section 18.15, LIN Self-Test Mode**. |

**Table 18.32 Operations Available in Each Mode**

| LIN Operation Mode | LIN Wake-Up Mode | LIN Self-Test Mode |
|---|---|---|
| Header transmission<br>Response transmission<br>Response reception<br>Error detection | Wake-up transmission<br>Wake-up reception<br>Error detection | self-test |

Whether a transition has been made to LIN reset mode, LIN operation mode, or LIN wake-up mode can be verified by reading the OMM1 and OMM0 bits in the RLN24nmLiMST register.

For a description of LIN self-test mode, see **Section 18.15, LIN Self-Test Mode**.

## 18.6 LIN Reset Mode

Setting the OM0 bit in the RLN24nmLiCUC register to 0 (LIN reset mode) causes a transition to LIN reset mode. The transition to LIN reset mode can be verified by determining that the OMM0 bit in the RLN24nmLiMST register has been set to 0 (LIN reset mode). In this mode, the LIN communication stops.

From LIN reset mode, transitions to LIN operation mode, LIN wake-up mode, and LIN self-test mode can be made.

When the mode transitions to LIN reset mode, the following registers are initialized to their reset values and retain their initial values while in LIN reset mode:

- RLN24nmLiTRC register
- RLN24nmLiST register
- RLN24nmLiEST register

The following registers retain their previous values even when a transition to LIN reset mode is made:

- RLN24nGLWBR register
- RLN24nGLBRP0 register
- RLN24nGLBRP1 register
- RLN24nmLiMD register
- RLN24nmLiBFC register
- RLN24nmLiSC register
- RLN24nmLiWUP register
- RLN24nmLiIE register
- RLN24nmLiEDE register
- RLN24nmLiDFC register
- RLN24nmLiIDB register
- RLN24nmLiCBR register
- RLN24nmLiDBRb register

## 18.7 LIN Operation Mode

While in LIN operation mode, frame processing (header transmission, response transmission, response reception, and error detection) is performed.

During a transition from LIN reset mode to LIN mode, setting the OM1 and OM0 bits in the RLN24nmLiCUC register to $11_B$ changes the mode to LIN operation mode, changing the OMM1 and OMM0 bits in the RLN24nmLiMST register to $11_B$. Communication settings should be performed after the RLN24nmLiMST register has become $11_B$.

## 18.8 LIN Wake-Up Mode

In LIN wake-up mode, wake-up signal processing (wake-up transmission, wake-up reception, and error detection) is performed.

During a transition from LIN reset mode to LIN mode, setting the OM1 and OM0 bits in the RLN24nmLiCUC register to $01_B$ changes the mode to LIN wake-up mode, changing the OMM1 and OMM0 bits in the RLN24nmLiMST register to $01_B$. Communication settings should be performed after the RLN24nmLiMST register has become $01_B$.

# 18.9 Header Transmission/Response Transmission/Response Reception

## 18.9.1 Header Transmission

**Figure 18.4** shows the operation of the LIN master interface in header transmission. **Table 18.33** shows processing in header transmission.

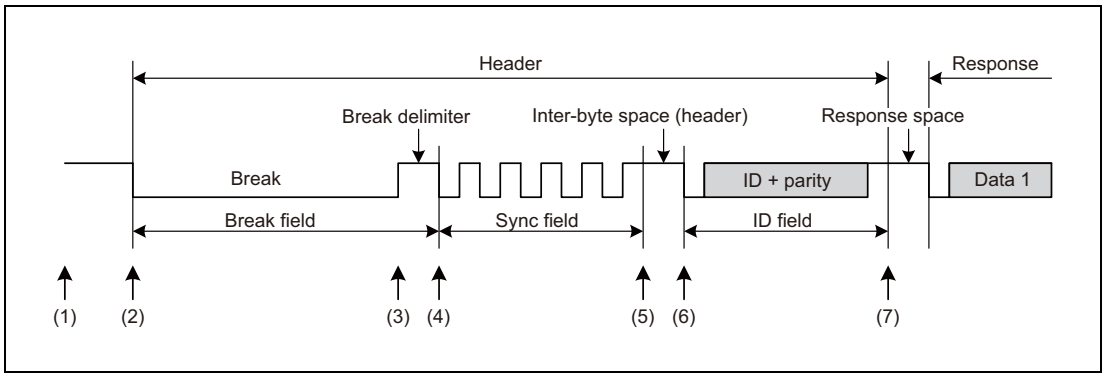



**Figure 18.4 Operation in Header Transmission**

**Table 18.33 Processing in Header Transmission**

| | Software Processing | LIN Master Interface Processing |
|---|---|---|
| (1) | • Sets a baud rate<br>• Enables interrupts<br>• Enables error detection<br>• Sets frame configuration parameters<br>• Transitions to LIN operation mode<br>• Sets information on the frame to be transmitted (ID, parity, data length, response direction, checksum method, and transmission data) | Waits for the FTS bit in the RLN24nmLiTRC register to be set by software (idle). |
| (2) | Sets the FTS bit in the RLN24nmLiTRC register to 1 (frame transmission or wake-up transmission/reception started) | Transmits a break. |
| (3) | Waits for an interrupt request | Transmits a break delimiter. |
| (4) | | Transmits a sync field ($55_H$). |
| (5) | | Transmits an inter-byte space (header). |
| (6) | | Transmits an ID field. |
| (7) | | Sets the successful header transmission flag. |

**NOTE**

For information about error detection, see **Section 18.14, Error Status**.

### 18.9.2 Response Transmission

**Figure 18.5** shows the operation of the LIN master interface in response transmission. **Table 18.34** shows the processing in response transmission.

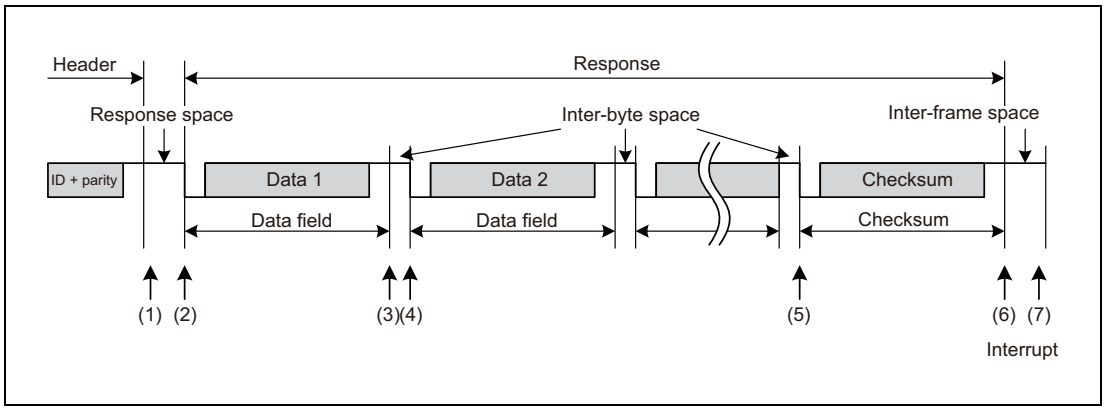



**Figure 18.5 Operation in Response Transmission**

**Table 18.34 Processing in Response Transmission**

| | Software Processing | LIN Master Interface Processing |
|---|---|---|
| (1) | (When in frame separate mode)<br>• Sets the RTS bit in the RLN24nmLiTRC register to 1 (response transmission started)<br>(When not in frame separate mode)<br>• Waits for an interrupt request | (When in frame separate mode)<br>• Waits for the RTS bit in the RLN24nmLiTRC register to be set to 1 by software. (During this time, “1” is output.)<br>• When the bit is set to 1, sends a response space.<br>(When not in frame separate mode)<br>• Sends a response space. |
| (2) | Waits for an interrupt request | Transmits data 1. |
| (3) | | Transmits an inter-byte space. |
| (4) | | • Transmits data 2.<br>• Transmits an inter-byte space<br>• Transmits data 3.<br>• Transmits an inter-byte space<br>(Repeats the transmission of data and inter-byte spaces as many times as the data length specified in bits RFDL[3:0] in the RLN24nmLiDFC register.)<br>:<br>: |
| (5) | | Transmits the checksum. |
| (6) | | • Sets the successful frame/wake-up transmission flag.<br>• Sets the FTS bit in the RLN24nmLiTRC register to 0 (frame transmission or wake-up transmission/reception stopped)<br>(When in frame separate mode)<br>• Sets the RTS bit in the RLN24nmLiTRC register to 0 (response transmission stopped). |
| (7) | • Processing after communication Checks the RLN24nmLiST register, and clears flags. | Idle |

**NOTE**

For information about error detection, see **Section 18.14, Error Status**.

### 18.9.3 Response Reception

**Figure 18.6** shows the operation of the LIN master interface in response reception. **Table 18.35** shows the processing in response reception.

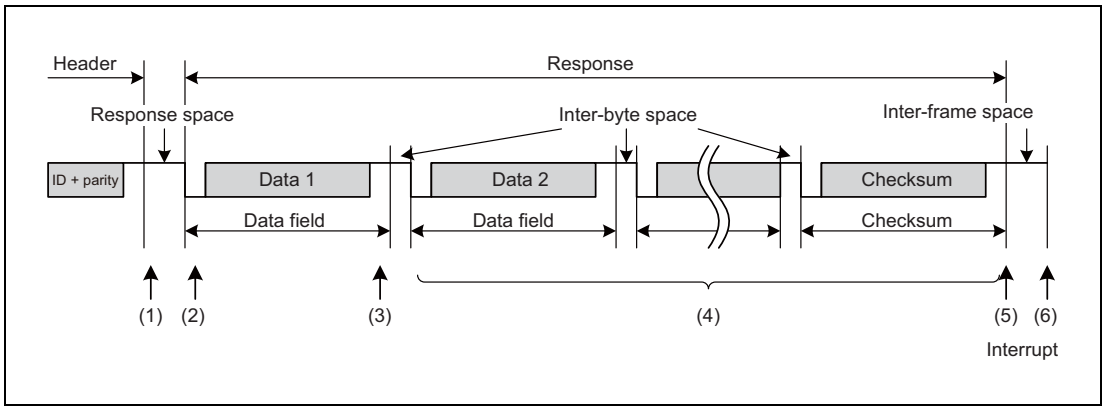



**Figure 18.6 Operation in Response Reception**

**Table 18.35 Processing in Response Reception**

| | Software Processing | LIN Master Interface Processing |
|---|---|---|
| (1) | Waits for an interrupt request (no processing). | Waits for detection of a start bit. |
| (2) | Waits for an interrupt request. | Receives data 1 when the start bit is detected. |
| (3) | | Sets the successful data 1 reception flag. |
| (4) | | • Receives data 2 when the start bit is detected.<br>• Receives data 3 when the start bit is detected.<br>(Repeats the reception of data as many times as the data length specified in bits RFDL[3:0] in the RLN24nmLiDFC register.)<br>⋮<br>• Receives the checksum when the start bit is detected. |
| (5) | | • Evaluates the checksum.<br>• Sets the successful frame/wake-up reception flag.<br>• Sets the FTS bit in the RLN24nmLiTRC register to 0 (frame transmission or wake-up transmission/reception stopped). |
| (6) | • Processing after communication<br>Reads the received data.<br>Checks the RLN24nmLiST register, and clears flags. | Idle |

**NOTE**

For information about error detection, see **Section 18.14, Error Status**.

# 18.10 Data Transmission/Reception

## 18.10.1 Data Transmission

One bit of data is transmitted per 1 Tbit.

The data that is transmitted returns to the reception data input pin via the LIN transceiver. The received data and the transmitted data are compared bit by bit, and the results are stored in the BER flag of the RLN24nmLiEST register (see **Section 18.14, Error Status**).

In the LIN master interface, because data is generated every 1 Tbit = $16f_{LIN}$, the sampling point for received data is the 13th clock cycle (81.25% position).

**Figure 18.7** shows an example of data transmission timing.

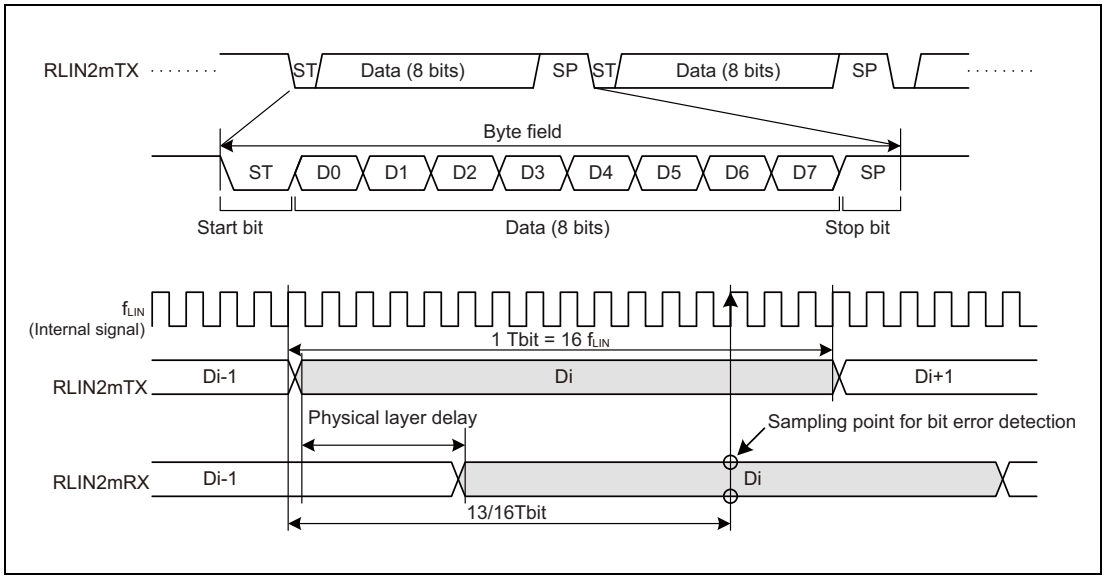



**Figure 18.7 Example of Data Transmission Timing**

### 18.10.2 Data Reception

Data reception is performed by using the synchronized RLIN2mRX signal (an internal signal) that is the input from the RLIN2mRX pin synchronized with the LIN system clock ($f_{LIN}$).

The byte field is synchronized at the falling edge of the start bit for the synchronized RLIN2mRX signal. After the falling edge is detected, sampling is performed again 0.5 Tbit later, and the falling edge is recognized as a start bit if the synchronized RLIN2mRX signal is low level. The falling edge is not recognized as a start bit if the RLIN2mRX signal after the reset is de-asserted is fixed to low level or if a high level is detected on re-sampling.

The bit sampling period after detection of the start bit is one Tbit.

**Figure 18.8** shows an example of data reception timing.

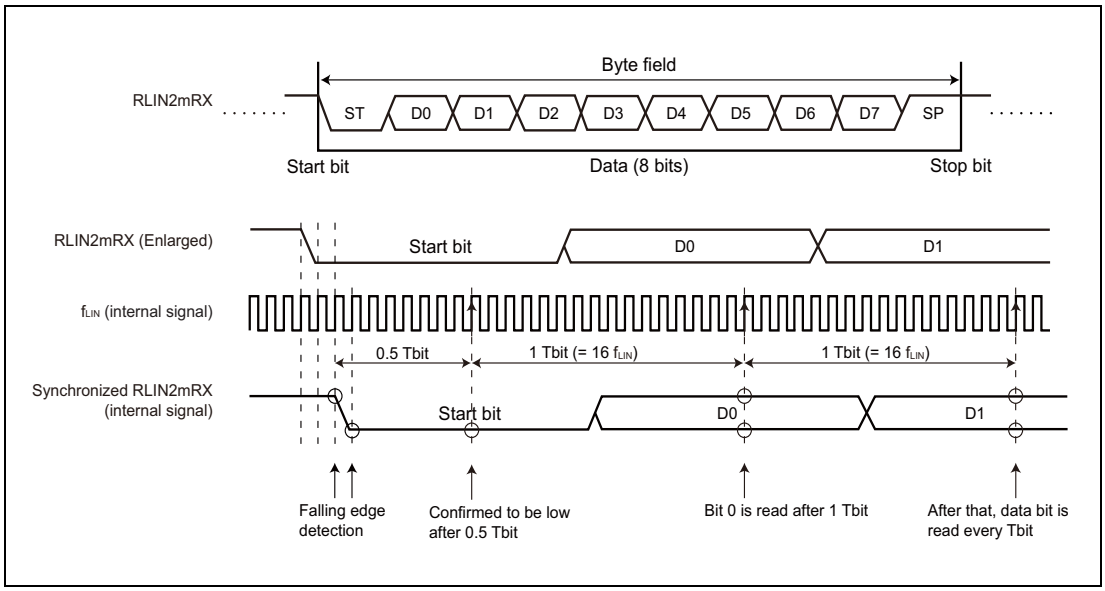



**Figure 18.8 Example of Data Reception Timing**

# 18.11 Transmission/Reception Data Buffering

This section explains the buffer processing that takes place when the LIN master interface sends or receives data continuously.

## 18.11.1 Transmission of LIN Frames

For an 8-byte transmission, the contents stored in registers RLN24nmLiDBR1 to RLN24nmLiDBR8 are sequentially transmitted to data areas 1 to 8 of the LIN frame. In the case of a 4-byte transmission, the contents stored in registers RLN24nmLiDBR1 to RLN24nmLiDBR4 are transmitted to data areas 1 to 4 of the LIN frame, but the contents of registers RLN24nmLiDBR5 to RLN24nmLiDBR8 are not transmitted. The transmitted checksum data is stored in the RLN24nmLiCBR register.

**Figure 18.9** shows the LIN transmission processing and the corresponding buffers.

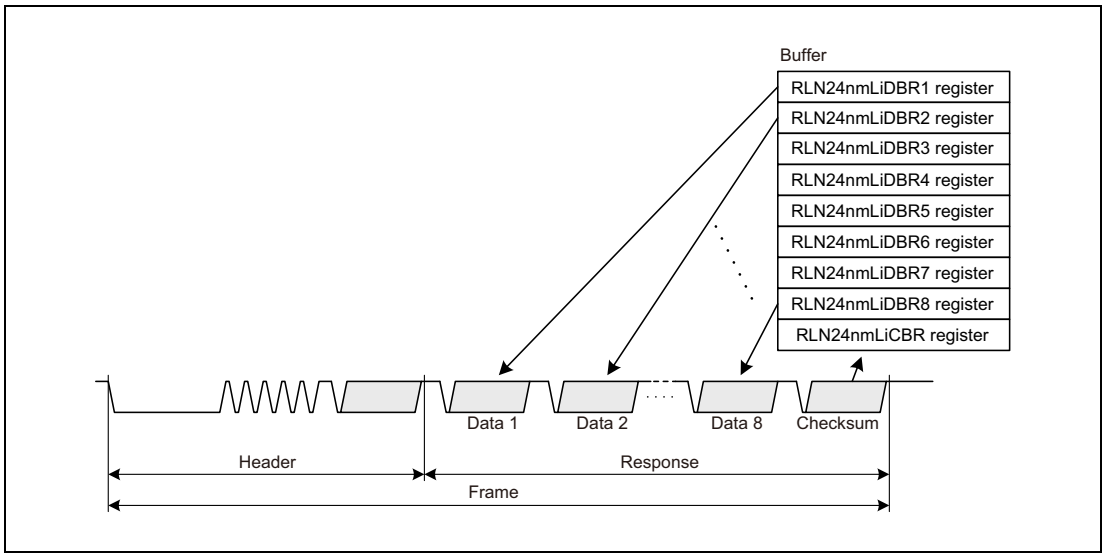



**Figure 18.9 LIN Transmission Processing and Corresponding Buffers**

### (1) Frame Separate Mode

Setting the FSM bit in the RLN24nmLiDFC register to 1 sets the frame separate mode.

In frame separate mode, a header and a response are transmitted when prompted by separate transmission start requests.

When the transmission of a header is finished, the HTRC flag in the RLN24nmLiST register is set to 1 (successful header transmission).

### 18.11.2 Reception of LIN Frames

For an 8-byte reception, the contents of data areas 1 to 8 of the LIN frame are stored in registers RLN24nmLiDBR1 to RLN24nmLiDBR8, respectively, upon reception of a stop bit. In the case of a 4-byte reception, the contents of data areas 1 to 4 of the LIN frame are stored in registers RLN24nmLiDBR1 to RLN24nmLiDBR4, respectively; no data is stored in registers RLN24nmLiDBR5 to RLN24nmLiDBR8. The received checksum data is stored in the RLN24nmLiCBR register.

**Figure 18.10** shows the LIN reception processing and the corresponding buffers.

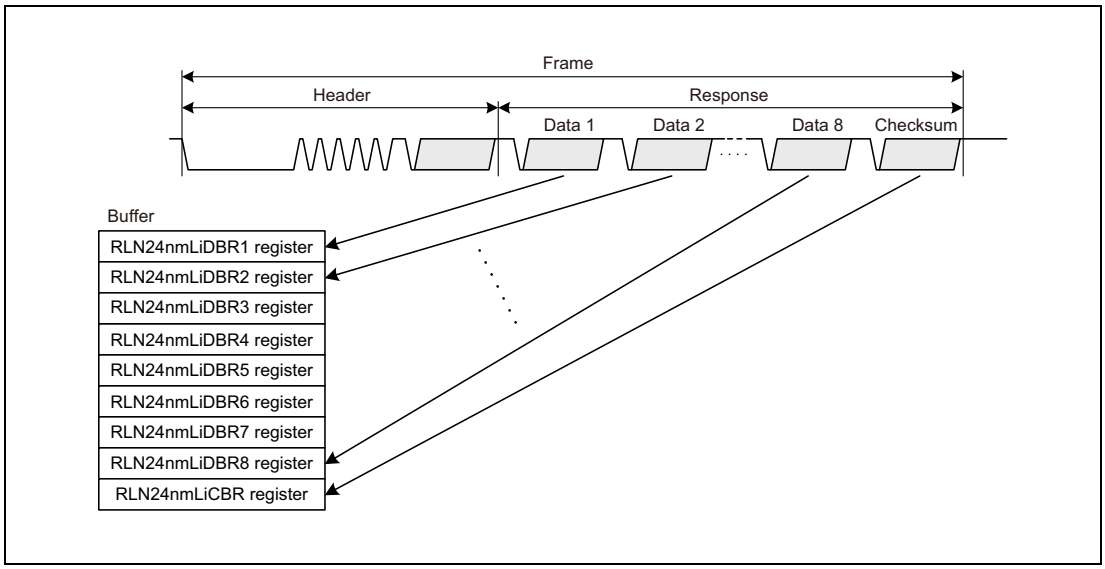



**Figure 18.10 LIN Reception Processing and Corresponding Buffers**

#### (1) Reception of Data 1

When the reception of the first byte of data is finished, the D1RC flag in the RLN24nmLiST register is set to 1 (successful data 1 reception).

## 18.12 Wake-Up Transmission/Reception

The wake-up transmission/reception can be used in LIN wake-up mode.

### 18.12.1 Wake-Up Transmission

In LIN wake-up mode, setting the RFT bit in the RLN24nmLiDFC register to 1 (transmission) and the FTS bit in the RLN24nmLiTRC register to 1 (frame transmission or wake-up transmission/reception started) causes a wake-up signal to be output from the output pin. The low level width of the wake-up signal is set using the WUTL[3:0] bits in the RLN24nmLiWUP register. However, if the value of the LWBR0 bit of the RLN24nGLWBR register is 1 (when LIN 2.x is used), the LIN system clock ($f_{LIN}$) has the low-level width of fa (JW) regardless of the setting of the LCKS bit of the RLN24nmLiMD register. By setting the baud rate to 19200 bps when fa is selected and setting the WUTL[3:0] bits of the RLN24nmLiWUP register to $0100_B$ (5 Tbits), 260 µs low level width can be output in LIN wake-up mode regardless of the setting of the LCKS bit of the RLN24nmLiMD register.

If a wake-up low level is output without any error, the FTC flag in the RLN24nmLiST register is set to 1 (successful frame or wake-up transmission); when the FTCIE bit in the RLN24nmLiIE register is 1 (successful frame/wake-up transmission interrupt enabled), an interrupt request is generated.

If an error is detected, wake-up transmission is aborted and the error flag for the error detected (the PBER flag or BER flag in the RLN24nmLiEST register) is set to 1 (physical bus error detection / bit error detection).

**Figure 18.11** shows the wake-up transmission timing.

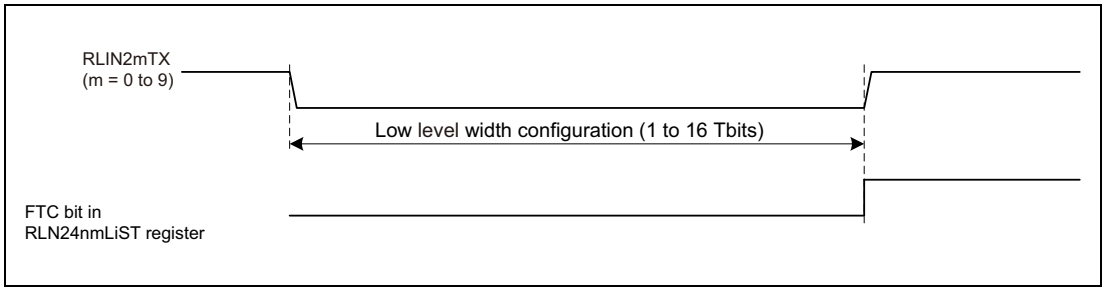



**Figure 18.11 Wake-Up Transmission Timing**

### 18.12.2 Wake-Up Reception

The detection of a wake-up signal involves the use of an input signal low level width count function.

The input signal low level width count function measures the low level width of the input signal to the RLIN2mRX pin, using the same sampling point as data reception. This allows an input signal of $f_{LIN}$ with a low-level width of 2.5-Tbit or longer to be measured.

When LIN Specification Package Revision 1.3 is used, set the LWBR0 bit in the RLN24nGLWBR register to 0. When LIN Specification Package Revision 2.x is used, set LWBR0 bit to 1.

When LWBR0 bit is set to 1, regardless of the setting of the LCKS bit in the RLN24nmLiMD register, fa is selected as the LIN system clock ($f_{LIN}$) (the LCKS bit is not changed).

Setting the baud rate to 19200 bps while fa is selected allows an input signal with a low-level width of 130 µs or longer to be detected during LIN wake-up mode regardless of the setting of the LCKS bit in the RLN24nmLiMD register.

When using this function, in LIN wake-up mode set the RFT bit in the RLN24nmLiDFC register to 0 (reception), and the FTS bit in the RLN24nmLiTRC register to 1 (frame transmission or wake-up transmission/reception started).

When the low level width to be measured is reached, the FRC flag in the RLN24nmLiST register is set to 1 (successful frame or wake-up reception) and if the FRCIE bit in the RLN24nmLiIE register is 1 (successful frame or wake-up reception interrupt enabled), an interrupt request is generated.

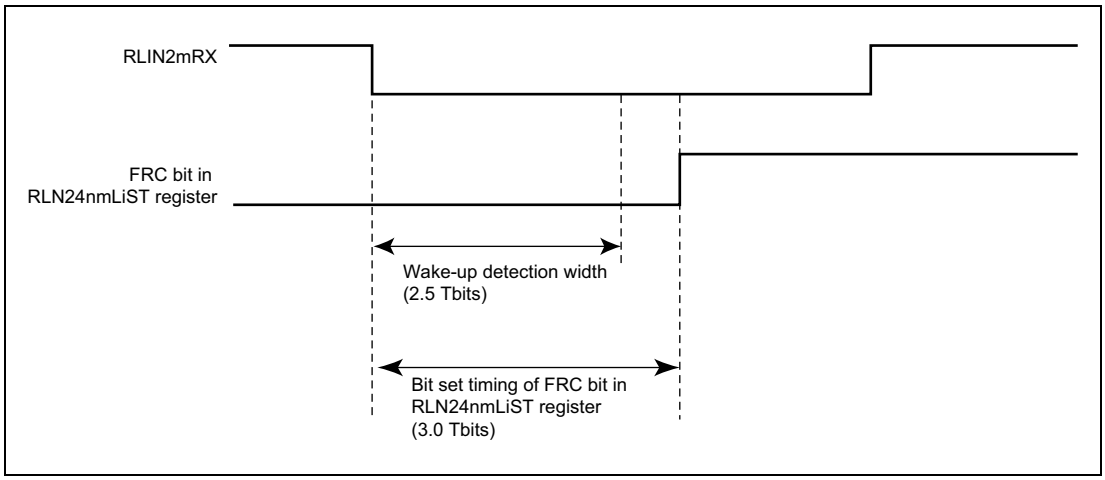



**Figure 18.12 Input Signal Low Level Count Function**

### 18.12.3 Wake-Up Collision

If the master node and the slave node transmit wake-up signals simultaneously, a collision will occur on the LIN bus; however the a collision of wake-up signals is not detected in the LIN master interface.

## 18.13 Status

During LIN mode operation, the LIN master interface can detect seven types of statuses.

Three of these statuses, successful frame/wake-up transmission, successful frame/wake-up reception, error detection, can generate interrupt requests.

**Table 18.36** shows the types of statuses.

**Table 18.36 Types of Statuses**

| Status | Status Set Condition | Status Clear Condition | Operation Mode Capable of Status Detection | Corresponding Bit | Interrupt |
|---|---|---|---|---|---|
| Reset | After the OM0 bit in the RLN24nmLiCUC register is set to not-LIN-reset-mode, if the LIN master interface actually exits LIN reset mode. | After the OM0 bit in the RLN24nmLiCUC register is set to LIN reset mode, if the LIN master interface enters LIN reset mode. | All modes | OMM0 bit in the RLN24nmLiMST register | — |
| Operation mode | After the OM1 bit in the RLN24nmLiCUC register is set to LIN operation mode, if the LIN master interface actually enters LIN operation mode. | After the OM1 bit in the RLN24nmLiCUC register is set to LIN wake-up mode, if the LIN master interface enters LIN wake-up mode. | • LIN operation mode<br>• LIN wake-up mode | OMM1 bit in the RLN24nmLiMST register | — |
| Successful frame/wake-up transmission | When a frame (header transmission + response transmission) or a wake-up signal is transmitted successfully. | • When the next communication is started<br>• When cleared by software<br>• After transition to LIN reset mode | • LIN operation mode<br>• LIN wake-up mode | FTC flag in the RLN24nmLiST register | √ |
| Successful frame/wake-up reception | When a frame (header transmission + response reception) or a wake-up signal is received successfully. | • When the next communication is started<br>• When cleared by software<br>• After transition to LIN reset mode | • LIN operation mode<br>• LIN wake-up mode | FRC flag in the RLN24nmLiST register | √ |
| Error detection | If any of the CSER flag, FER flag, FTER flag, PBER flag, and BER flag in the RLN24nmLiEST register is set to 1 (error detected). | • When the next communication is started<br>• When cleared by software*1<br>• After transition to LIN reset mode | • LIN operation mode<br>• LIN wake-up mode | ERR flag in the RLN24nmLiST register | √ |
| Successful data 1 reception | When the RFT bit in the RLN24nmLiDFC register is 0 (reception) and the first byte of the response field is received successfully.*2 | • When the next communication is started<br>• When cleared by software<br>• After transition to LIN reset mode | LIN operation mode | D1RC flag in the RLN24nmLiST register | — |
| Successful header transmission | When a header field is transmitted successfully. | • When the next communication is started<br>• When cleared by software<br>• After transition to LIN reset mode | LIN operation mode | HTRC flag in the RLN24nmLiST register | — |

Note 1. While in LIN operation mode, the ERR flag in the RLN24nmLiST register is cleared to 0 by writing 0 to the CSER flag, FER flag, FTER flag, PBER flag, or BER flag in the RLN24nmLiEST register.

Note 2. Not detected when the RFDL [3:0] bits in the RLN24nmLiDFC register are $0000_B$ (0 bytes + checksum).

# 18.14 Error Status

## 18.14.1 Types of Error Statuses

The LIN master interface can detect five types of error statuses in LIN master mode. The condition of these error statuses can be checked by means of the corresponding bits in the RLN24nmLiEST register.

All error statuses represent interrupt sources.

**Table 18.37** shows the types of error statuses.

**Table 18.37 Types of Error Statuses**

| Status | Error Detection Condition | Operating Mode Capable of Error Detection | Communication | Selection of Detection Enable/ Disable | Corresponding Bit |
|---|---|---|---|---|---|
| Bit error | The transmitted data and the data on the LIN bus monitored by the receive pin do not match *1 | • LIN operation mode<br>• LIN wake-up mode | Aborted | √ | BER flag in the RLN24nmLiEST register |
| Physical bus error | • LIN bus is detected to be high level when sending a break<br>• LIN bus is detected to be low level when sending a break delimiter<br>• LIN bus is detected to be high level when sending a wake-up | • LIN operation mode<br>• LIN wake-up mode | Aborted | √ | PBER flag in the RLN24nmLiEST register |
| Frame timeout error | A frame transmission/reception does not complete within a given time*2 | LIN operation mode | Aborted | √ | FTER flag in the RLN24nmLiEST register |
| Framing error | In response field reception, the stop bit of each data byte is low level | LIN operation mode | Aborted | √ | FER flag in the RLN24nmLiEST register |
| Checksum error | In response field reception, checksum test results in an error | LIN operation mode | — | × | CSER flag in the RLN24nmLiEST register |

Note 1. If a bit error is detected, the process is aborted after a stop bit is sent. If a bit error is detected in a non-data area, such as an inter-byte space, the transmission is aborted immediately. If a bit error is detected during the transmission of a wake-up, the transmission of the wake-up is aborted after the bit that caused the error is transmitted.

Note 2. The timeout time depends on the response field data length (the RFDL[3:0] bits in the RLN24nmLiDFC register) and the checksum selection (the CSM bit in the RLN24nmLiDFC register), and this can be calculated according to the following formula:
When classic checksum is selected (when the CSM bit in the RLN24nmLiDFC is 0): Timeout time = 49 + (number of data bytes + 1) × 14 [Tbit]
When enhanced checksum is selected (when the CSM bit in the RLN24nmLiDFC is 1): Timeout time = 48 + (number of data bytes + 1) × 14 [Tbit]

The aforementioned timeout time is longer than the TFRAME_MAX of LIN Specification Package Revision 1.3 when classic checksum is selected, or the TFRAME_MAX of LIN Specification Package Revision 2.x when enhanced checksum is selected.

The error status is cleared when the next communication is started, when it is cleared by software, or at a transition to LIN reset mode.

## 18.14.2 Target Time Domain for Error Detection

**Figure 18.13** shows the time domain in which the LIN master interface monitors for error detection.

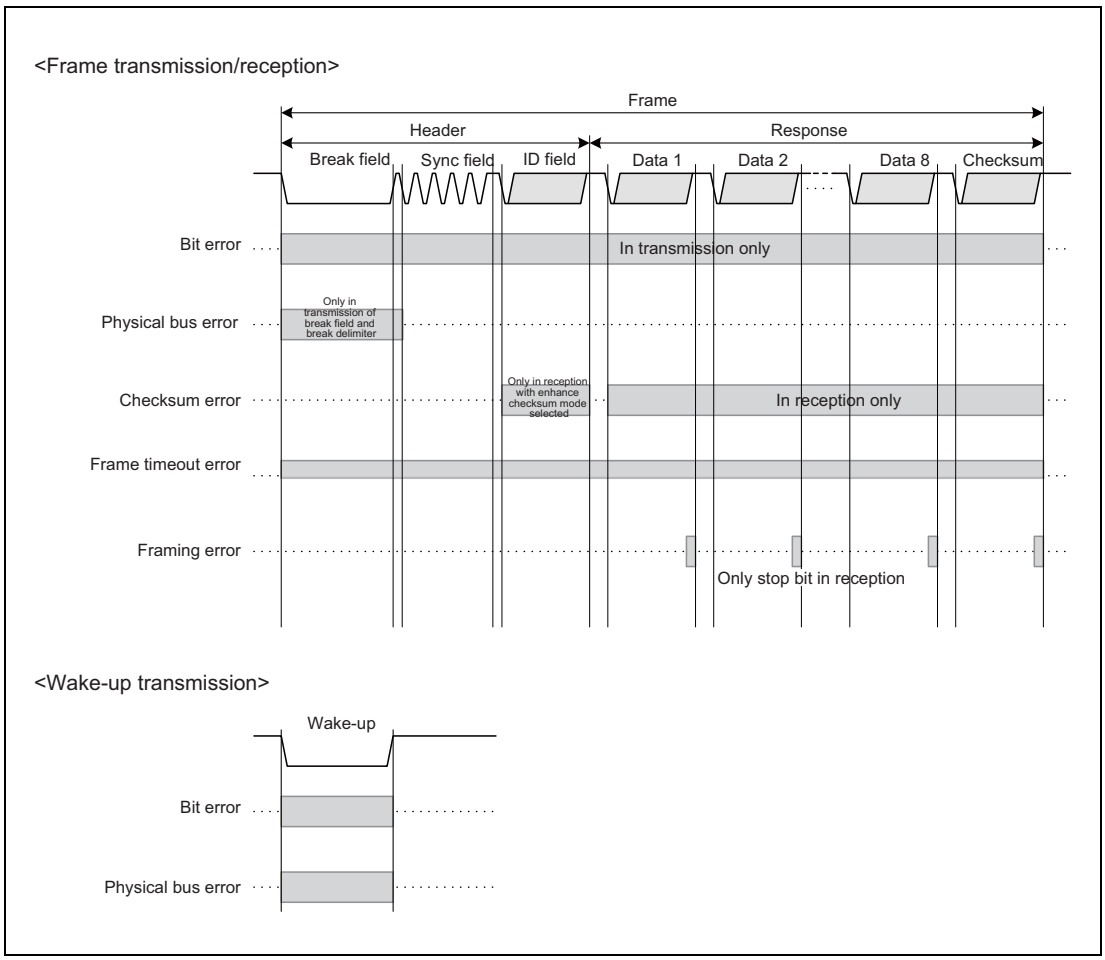



**Figure 18.13 Target Time Domain for Error Detection**

## 18.15 LIN Self-Test Mode

The LIN master interface provides LIN self-test mode. When the LIN master interface enters LIN self-test mode, RLIN2mTX and RLIN2mRX are disconnected from the external pins, and are internally connected in the LIN master interface. Thus, the frame transmitted from RLIN2mTX is looped back to RLIN2mRX.

Two types of self-test can be performed:

- LIN self-test mode (transmission): header transmission and response transmission
- LIN self-test mode (reception): header transmission and response reception

In LIN self-test mode, the operation is performed at the fastest baud rate, regardless of the setting of the baud rate generator. Regardless of the setting of the baud rate related registers, the baud rate operates at the LIN communication clock source/16 [bps].

In LIN self-test mode, the following functions are not supported:

- LIN wake-up mode
- Frame separate mode

Do not use these functions.

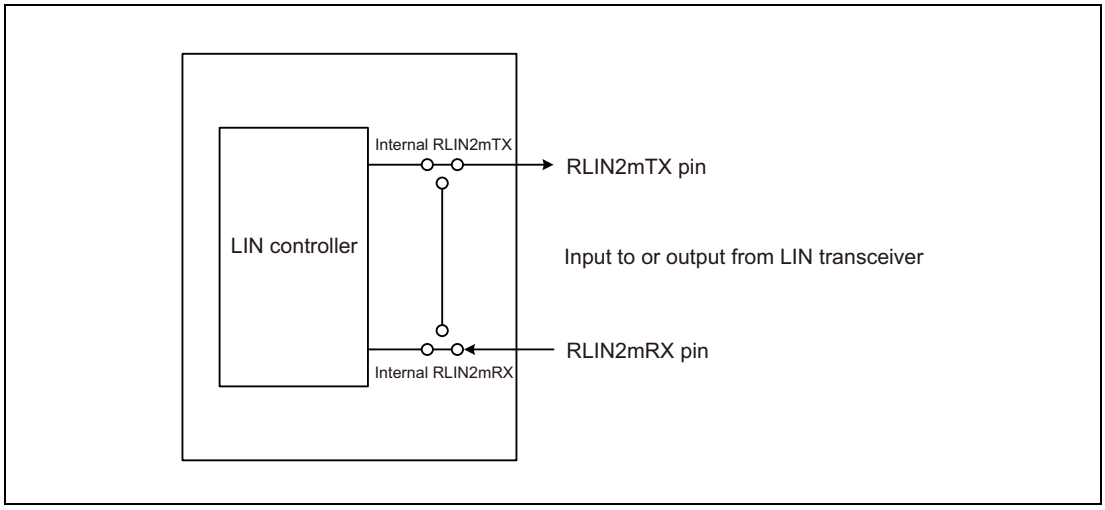



**Figure 18.14 Connection in LIN Reset Mode, LIN Wake-Up Mode, and LIN Operation Mode**

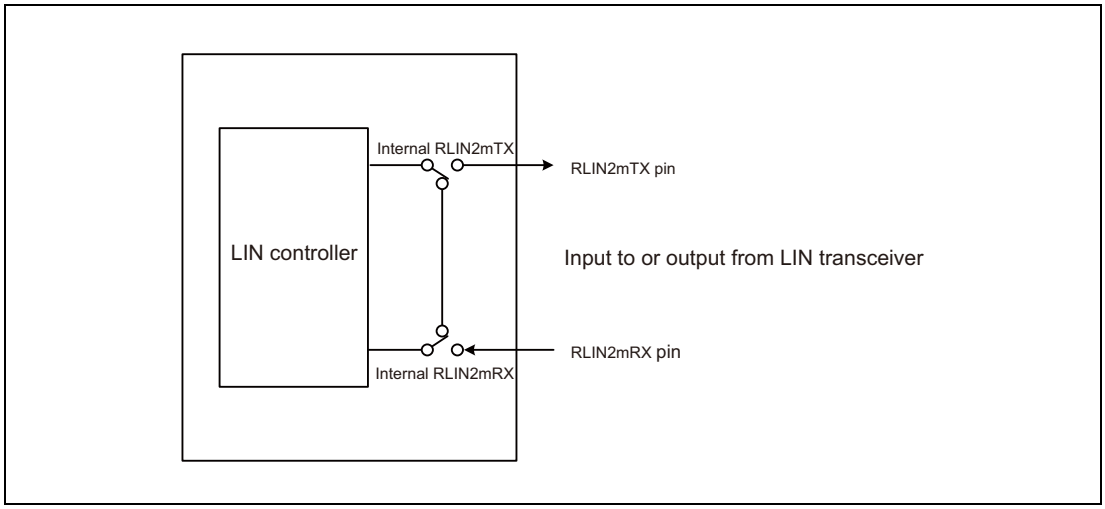



**Figure 18.15 Connection in LIN Self-Test Mode**

### 18.15.1 Transition to LIN Self-Test Mode

Writing to the RLN24nGLSTC register enables LIN self-test mode.

The LSTM bit in the RLN24nGLSTC register being set to 1 indicates that the mode has transitioned to LIN self-test mode.

A specific sequence is required to transition to LIN self-test mode. In this sequence, information must be written three times consecutively to the LIN self-test control register, as follows:

- Switch all channels of the unit to LIN reset mode.
  Set the OM0 bit in the RLN24nmLiCUC register to 0 (LIN reset mode).
  Read the OMM0 bit in the RLN24nmLiMST register and confirm that it is 0 (LIN reset mode).
- 1st write: RLN24nGLSTC register = $1010\ 0111_B$ ($A7_H$)
- 2nd write: RLN24nGLSTC register = $0101\ 1000_B$ ($58_H$)
- 3rd write: RLN24nGLSTC register = $0000\ 0001_B$ ($01_H$)
- Confirm that all channels have transitioned to LIN self-test mode
  Read the LSTM bit in the RLN24nGLSTC register; verify that it is 1 (LIN self-test mode).

If the key of the first write ($A7_H$) is written twice by mistake, the transition to LIN self-test mode is canceled. The above sequence should be retried from the 1st write step. In addition, if a write to another LIN-related register in the same unit is performed during transition to LIN self-test mode (three consecutive write operations to the RLN24nGLSTC register), the transition is also canceled.

### 18.15.2 Transmission in LIN Self-Test Mode

To execute a self-test on transmission, perform the procedure below:

- Set the baud rate related registers.
  RLN24nGLBRP0 register = xxxx $xxxx_B$ *1
  RLN24nGLBRP1 register = xxxx $xxxx_B$ *1
  RLN24nmLiMD register = 0000 $xx00_B$ *1
- Set interrupt enable register and error enable related registers.
  RLN24nmLiIE register = 0000 $0xxx_B$ *2
  RLN24nmLiEDE register = 0000 $xxxx_B$
- Set the break field and space related registers.
  RLN24nmLiBFC register = 00xx $xxxx_B$
  RLN24nmLiSC register = 00xx $0xxx_B$
- Exit the LIN reset mode.
  Write $11_B$ to the OM1 and OM0 bits in the RLN24nmLiCUC register, and check that the OMM1 and OMM0 bits in the RLN24nmLiMST register are $11_B$.
- Set the transmit frame related registers.
  RLN24nmLiDFC register = 00x1 $xxxx_B$
  RLN24nmLiIDB register = xxxx $xxxx_B$
  RLN24nmLiDBR1 to RLN24nmLiDBR8 registers = xxxx $xxxx_B$
- Start header transmission → response transmission
  Set the FTS bit in the RLN24nmLiTRC register to 1 (frame transmission or wake-up transmission/reception started).
  The LIN self-test mode (transmission) is executed, interrupts are generated, and status and error status are also updated. The checksum is automatically calculated by the LIN master interface.
  To suspend the LIN self-test mode (transmission) while it is running, set the OM0 bit in the RLN24nmLiCUC register to 0 (LIN reset mode), which causes a transition to the LIN reset mode.
- When the transmission is completed, the reversed value of the looped-back frame data is stored in the RLN24nmLiIDB, RLN24nmLiDBRb, and RLN24nmLiCBR registers (the data is reversed before being stored because the transmitted value should be compared with the looped-back value). Then, the FTS bit in the RLN24nmLiTRC register is cleared.
- If the transmission fails to complete due to an error, the applicable error flag is set and the FTS bit in the RLN24nmLiTRC register is cleared.

**NOTE**

x: Don't care

**Note 1.** The settings of the following registers are not reflected in the operation of the LIN self-test mode:
the RLN24nGLBRP0 register, the RLN24nGLBRP1 register, and the LCKS bit in the RLN24nmLiMD register. Therefore, setting these registers is not necessary.

**Note 2.** If necessary, set the related registers described in **Section 7, Exception/Interrupts**.

### 18.15.3 Reception in LIN Self-Test Mode

To execute a self-test on reception, perform the procedure below:

- Set the baud rate related registers.
  RLN24nGLBRP0 register = xxxx $xxxx_B$*1
  RLN24nGLBRP1 register = xxxx $xxxx_B$*1
  RLN24nmLiMD register = 0000 $xx00_B$*1
- Set the interrupt enable and error enable related registers.
  RLN24nmLiIE register = 0000 $0xxx_B$*2
  RLN24nmLiEDE register = 0000 $x0xx_B$
- Set the break field and space related registers.
  RLN24nmLiBFC register = 00xx $xxxx_B$
  RLN24nmLiSC register = 00xx $0xxx_B$*1
- Exit the LIN reset mode.
  Write $11_B$ to the OM1 and OM0 bits in the RLN24nmLiCUC register, and check that the OMM1 and OMM0 bits in the RLN24nmLiMST register are $11_B$.
- Set the receive frame related registers.
  RLN24nmLiDFC register = 00x0 $xxxx_B$
  RLN24nmLiIDB register = xxxx $xxxx_B$
  RLN24nmLiDBR1 to RLN24nmLiDBR8 registers = xxxx $xxxx_B$
  RLN24nmLiCBR register = xxxx $xxxx_B$
  Since the checksum value to be transmitted is not automatically calculated, perform the calculation and specify the calculated value in the RLN24nmLiCBR register. By specifying an incorrect checksum, the checksum error can be tested.
- Start header transmission → response reception
  Set the FTS bit in the RLN24nmLiTRC register to 1 (frame transmission or wake-up transmission/reception started).
  The LIN self-test mode (reception) is executed, interrupts are generated, and status and error status are also updated.
  To suspend the LIN self-test mode (reception) while it is running, set the OM0 bit in the RLN24nmLiCUC register to 0 (LIN reset mode), which causes a transition to the LIN reset mode.
- When the reception is completed, the reversed value of the looped-back frame data is stored in the RLN24nmLiIDB, RLN24nmLiDBRb, and RLN24nmLiCBR registers (the data is reversed before being stored because the set value should be compared with the looped-back value). Then, the FTS bit in the RLN24nmLiTRC register is cleared.
- If the reception fails to complete due to an error, the applicable error flag is set and the FTS bit in the RLN24nmLiTRC register is cleared.

**NOTE**

x: Don't care

**Note 1.** The settings of the following registers are not reflected to the operation of the LIN self-test mode:
the RLN24nGLBRP0 register, the RLN24nGLBRP1 register, the LCKS bit in the RLN24nmLiMD register, and the IBS bit and IBHS bit (response space only) in the RLN24nmLiSC register. Therefore, setting these registers is not necessary.

**Note 2.** If necessary, set the related registers described in **Section 7, Exception/Interrupts**.

### 18.15.4 Exiting LIN Self-Test Mode

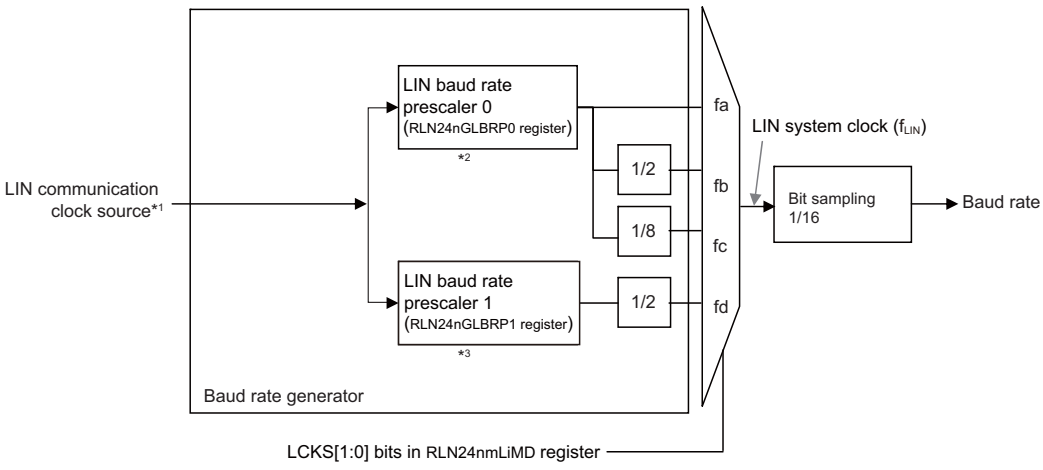
To exit LIN self-test mode, perform the procedure below:

- Switch all channels of the unit to LIN reset mode.
  Write 0 to the OM0 bit in the RLN24nmLiCUC register to make a transition to LIN reset mode. However, if the OMM1 and OMM0 bits in the RLN24nmLiMST register are not $11_B$ in any channels of the unit after the transition to LIN self-test mode, write $11_B$ to the OM1 and OM0 bits in the RLN24nmLiCUC register in any one channel. Check that the OMM1 and OMM0 bits in the RLN24nmLiMST register are set to $11_B$, and then make a transition to LIN reset mode.
- Verify that the LIN master interface has exited LIN self-test mode.
  Read the LSTM bit in the RLN24nGLSTC register and confirm that it is not 0 (not in LIN self-test mode)
- Verify the transition to LIN reset mode.
  Read the OMM0 bit in the RLN24nmLiMST register and confirm that it is 0 (LIN reset mode).

## 18.16 Baud Rate Generator

The LIN system clock ($f_{LIN}$) is obtained by dividing the LIN communication clock source frequency by the baud rate generator, and the baud rate is obtained by dividing that clock by 16. The inverse of this baud rate is called the bit time (Tbit).

**Figure 18.16** shows a block diagram of baud rate generation.




Note 1. For the LIN communication clock source, see **12.5.3, Clock Domain Settings**.
Note 2. According to the RLN24nGLBRP0 register, the clock frequency is divided by N + 1 (N = 0 to 255).
Note 3. According to the RLN24nGLBRP1 register, the clock frequency is divided by M + 1 (M = 0 to 255).

**Figure 18.16 Block Diagram of Baud Rate Generation**

Set the LIN communication clock source in a range from 4 MHz to 40 MHz.

By setting the RLN24nGLBRP0 register so that fa is 307200 Hz (= 19200 × 16), the resulting system clock frequencies are fa = 19200 × 16, fb = 9600 × 16, and fc = 2400 × 16. These system clock frequencies are divided by 16 in the bit timing generator, enabling baud rates of 19200 bps, 9600 bps, and 2400 bps to be generated. Also, by setting the RLN24nGLBRP1 register so that fd is 166672 Hz (= 10417 × 16), the resulting system clock frequency is fd = 10417 × 16. This system clock frequency is divided by 16 in the bit timing generator, enabling 10417 bps to be generated.

The formula for calculating baud rate is shown below.

*Baud rate*:

= {Frequency of LIN communication clock source} ÷ (RLN24nGLBRP0 + 1) ÷ 16 [bps] (When fa is selected)

= {Frequency of LIN communication clock source} ÷ (RLN24nGLBRP0 + 1) ÷ 2 ÷ 16 [bps] (When fb is selected)

= {Frequency of LIN communication clock source} ÷ (RLN24nGLBRP0 + 1) ÷ 8 ÷ 16 [bps] (When fc is selected)

= {Frequency of LIN communication clock source} ÷ (RLN24nGLBRP1 + 1) ÷ 2 ÷ 16 [bps] (When fd is selected)

# Section 19 LIN/UART Interface (RLIN3)

This section contains a generic description of the LIN/UART interface (RLIN3).

The first part of this section describes all RH850/F1K specific properties, such as the number of units, register base addresses, etc. The remainder of the section describes the functions and registers of RLIN3.

## 19.1 Features of RH850/F1K RLIN3

### 19.1.1 Number of Units and Channels

This microcontroller has the following number of RLIN3 units.

Each RLIN3 unit has a single channel interface. “Number of channels” therefore has the same meaning as “number of units” in this section.

**Table 19.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of units | 4 | 6 | 6 |
| Name | RLIN3n (n = 0 to 3) | RLIN3n (n = 0 to 5) | RLIN3n (n = 0 to 5) |

**Table 19.2 Unit Configurations and Channels**

| Unit Name (Channel Name) RLIN3n | Channels per Unit | RH850/F1K 100 pins (4 ch) | RH850/F1K 144 pins (6 ch) | RH850/F1K 176 pins (6 ch) |
|---|---|---|---|---|
| RLIN30 | 1 | √ | √ | √ |
| RLIN31 | 1 | √ | √ | √ |
| RLIN32 | 1 | √ | √ | √ |
| RLIN33 | 1 | √ | √ | √ |
| RLIN34 | 1 | | √ | √ |
| RLIN35 | 1 | | √ | √ |

**Note:** The channel names are same as those of the corresponding units.

**Table 19.3 Indices**

| Index | Description |
|---|---|
| n | Throughout this section, the individual RLIN3 units are identified by the index “n” (n = 0 to 5): for example, RLN3nLCUC is the LIN control register. |
| b | Throughout this section, the individual transmit/received data buffers of RLIN3n are identified by the index “b” (b = 1 to 8): for example, RLN3nLDBRb is the data buffer register. |

The following lists the index value corresponding to each product.

**Table 19.4 Index Correspondence of Each Product**

| Index Correspondence to Product |
|---|
| All products |
| b = 1 to 8 |

### 19.1.2 Register Base Address

RLIN3 base addresses are listed in the following table.

RLIN3 register addresses are given as offsets from the base addresses.

**Table 19.5 Register Base Addresses**

| Base Address Name | Base Address |
|---|---|
| <RLIN30_base> | FFCE 2000$_H$ |
| <RLIN31_base> | FFCE 2040$_H$ |
| <RLIN32_base> | FFCE 2080$_H$ |
| <RLIN33_base> | FFCE 20C0$_H$ |
| <RLIN34_base> | FFCE 2100$_H$ |
| <RLIN35_base> | FFCE 2140$_H$ |

### 19.1.3 Clock Supply

The RLIN3 clock supply is shown in the following table.

**Table 19.6 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| RLIN3n | LIN communication clock sources | CKSCLK_ILIN*1,*2 | Communication clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CKSCLK_ILIN | |

Note 1. The clock domain CKSCLK_ILIN divided clock can be supplied only to RLIN30 channel.

Note 2. Set the LIN communication clock source in the range of 4 MHz to 40 MHz.

### 19.1.4 Interrupt Request

RLIN3 interrupt requests are listed in the following table.

**Table 19.7 Interrupt Requests**

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **RLIN30** | | | |
| INTRLIN3n (n = 0) | RLIN30 interrupt | 33 | — |
| INTRLIN3nUR0 (n = 0) | RLIN30 transmit interrupt | 34 | 10 |
| INTRLIN3nUR1 (n = 0) | RLIN30 receive completion interrupt | 35 | 11 |
| INTRLIN3nUR2 (n = 0) | RLIN30 status interrupt | 36 | — |
| **RLIN31** | | | |
| INTRLIN3n (n = 1) | RLIN31 interrupt | 120 | — |
| INTRLIN3nUR0 (n = 1) | RLIN31 transmit interrupt | 121 | 86 |
| INTRLIN3nUR1 (n = 1) | RLIN31 receive completion interrupt | 122 | 87 |
| INTRLIN3nUR2 (n = 1) | RLIN31 status interrupt | 123 | — |
| **RLIN32** | | | |
| INTRLIN3n (n = 2) | RLIN32 interrupt | 164 | — |
| INTRLIN3nUR0 (n = 2) | RLIN32 transmit interrupt | 165 | 44 |
| INTRLIN3nUR1 (n = 2) | RLIN32 receive completion interrupt | 166 | 45 |
| INTRLIN3nUR2 (n = 2) | RLIN32 status interrupt | 167 | — |
| **RLIN33** | | | |
| INTRLIN3n (n = 3) | RLIN33 interrupt | 228 | — |
| INTRLIN3nUR0 (n = 3) | RLIN33 transmit interrupt | 229 | 111 |
| INTRLIN3nUR1 (n = 3) | RLIN33 receive completion interrupt | 230 | 112 |
| INTRLIN3nUR2 (n = 3) | RLIN33 status interrupt | 231 | — |
| **RLIN34** | | | |
| INTRLIN3n (n = 4) | RLIN34 interrupt | 232 | — |
| INTRLIN3nUR0 (n = 4) | RLIN34 transmit interrupt | 233 | 50 |
| INTRLIN3nUR1 (n = 4) | RLIN34 receive completion interrupt | 234 | 51 |
| INTRLIN3nUR2 (n = 4) | RLIN34 status interrupt | 235 | — |
| **RLIN35** | | | |
| INTRLIN3n (n = 5) | RLIN35 interrupt | 236 | — |
| INTRLIN3nUR0 (n = 5) | RLIN35 transmit interrupt | 237 | 113 |
| INTRLIN3nUR1 (n = 5) | RLIN35 receive completion interrupt | 238 | 114 |
| INTRLIN3nUR2 (n = 5) | RLIN35 status interrupt | 239 | — |

### 19.1.5 Reset Sources

RLIN3 reset sources are listed in the following table. RLIN3 is initialized by these reset sources.

**Table 19.8 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| RLIN3n | All reset sources (ISORES) |

### 19.1.6 External Input/output Signals

External input/output signals of RLIN3 are listed below.

**Table 19.9 External Input/Output Signals**

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **RLIN30** | | |
| RLIN3nRX (n = 0) | RLIN30 receive data input | RLIN30RX |
| RLIN3nTX (n = 0) | RLIN30 transmit data output | RLIN30TX |
| **RLIN31** | | |
| RLIN3nRX (n = 1) | RLIN31 receive data input | RLIN31RX |
| RLIN3nTX (n = 1) | RLIN31 transmit data output | RLIN31TX |
| **RLIN32** | | |
| RLIN3nRX (n = 2) | RLIN32 receive data input | RLIN32RX |
| RLIN3nTX (n = 2) | RLIN32 transmit data output | RLIN32TX |
| **RLIN33** | | |
| RLIN3nRX (n = 3) | RLIN33 receive data input | RLIN33RX |
| RLIN3nTX (n = 3) | RLIN33 transmit data output | RLIN33TX |
| **RLIN34** | | |
| RLIN3nRX (n = 4) | RLIN34 receive data input | RLIN34RX |
| RLIN3nTX (n = 4) | RLIN34 transmit data output | RLIN34TX |
| **RLIN35** | | |
| RLIN3nRX (n = 5) | RLIN35 receive data input | RLIN35RX |
| RLIN3nTX (n = 5) | RLIN35 transmit data output | RLIN35TX |

## 19.2 Overview

### 19.2.1 Functional Overview

The LIN/UART interface is a hardware LIN communication controller that supports LIN Specification Package Revision 1.3, 2.0, 2.1, 2.2, and SAE J2602, and automatically performs frame communication and error determination.

The LIN/UART interface is provided with UART mode and can also be used as a UART.

The appropriate mode should be used for the LIN/UART interface according to the application: LIN master, LIN slave, or UART.

**LIN master**

- LIN reset mode
- LIN mode (LIN master mode)
  - LIN wake-up mode
  - LIN operation mode
- LIN self-test mode

**LIN slave**

- LIN reset mode
- LIN mode (LIN slave mode [auto baud rate] or LIN slave mode [fixed baud rate])
  - LIN wake-up mode
  - LIN operation mode
- LIN self-test mode

**UART**

- LIN reset mode
- UART mode

**Table 19.10** shows the LIN/UART interface specifications.

**Table 19.10 LIN/UART Interface Specifications (1/3)**

| Item | | Specifications | |
|---|---|---|---|
| | Channel count | Up to 6 channels | |
| LIN communication function | Protocol | LIN Specification Package Revision 1.3, 2.0, 2.1, 2.2, and SAE J2602 | |
| | Variable frame structure | Master | • Break transmission width: 13 to 28 Tbits<br>• Break delimiter transmission width: 1 to 4 Tbits<br>• Transmission inter-byte space width (header): 0 to 7 Tbits (space between Sync field and ID field)*1<br>• Transmission response space width: 0 to 7 Tbits*1<br>• Transmission inter-byte space width: 0 to 3 Tbits (space between data bytes in response area)<br>• Transmission wake-up width: 1 to 16 Tbits |
| | | Slave | • Break reception width : 9.5 or 10.5 Tbits [for fixed baud rate]<br>: 10 or 11 Tbits [for auto baud rate]<br>• Transmission response space width: 0 to 7 Tbits<br>• Transmission inter-byte space width: 0 to 3 Tbits (space between data bytes in response area)<br>• Transmission wake-up width: 1 to 16 Tbits |
| | Checksum | • Automatic operation for both transmission and reception<br>• Classic or enhanced selectable (for each frame) | |
| | Response field data byte count | Variable from 0 to 8 bytes<br>Multi-byte (9 or more bytes) response transmission and reception also possible | |
| | Frame communication modes | Master | • Mode in which header transmission and response transmission/reception are started with a single transmission start request<br>• Mode in which header transmission and response transmission are started with separate transmission start requests (frame separate mode) |
| | | Slave | • Mode in which header is automatically received with fixed baud rate<br>• Mode in which header is automatically received with the baud rate set according to the sync field measurement result of the sync field and break field detected |
| | Wake-up transmission and reception | LIN wake-up mode provided<br>• Wake-up transmission (1 to 16 Tbits)<br>• Wake-up reception<br>Low-level width of input signals measured | |
| | Status | Master | • Successful frame/wake-up transmission<br>• Successful header transmission<br>• Successful frame/wake-up reception*2<br>• Successful data 1 reception<br>• Error detection<br>• Operation mode<br>(LIN reset mode, LIN wake-up mode, LIN operation mode, LIN self-test mode) |
| | | Slave | • Successful response/wake-up transmission<br>• Successful response/wake-up reception*2<br>• Successful header reception<br>• Successful data 1 reception<br>• Error detection<br>• Operation mode<br>(LIN reset mode, LIN wake-up mode, LIN operation mode, LIN self-test mode) |

Table 19.10 LIN/UART Interface Specifications (2/3)

| Item | | Specifications | |
|---|---|---|---|
| LIN communication function | Error status | Master | • Bit error<br>• Checksum error<br>• Frame timeout error/response timeout error<br>• Physical bus error<br>• Framing error<br>• Response preparation error |
| | | Slave | • Bit error<br>• Checksum error<br>• Frame timeout error/response timeout error<br>• Sync field error<br>• ID parity error<br>• Framing error<br>• Response preparation error |
| | Baud rate selection | Baud rates conforming to the LIN specifications generated using baud rate generator | |
| | Test mode | Self-test mode for user evaluation | |
| | Interrupt function | Master | • Successful header/frame/wake-up transmission<br>• Successful frame/wake-up reception*2<br>• Error detection |
| | | Slave | • Successful response/wake-up transmission<br>• Successful Header/response/wake-up reception*2<br>• Error detection |
| UART communication function | Data buffer | • Transmission data buffer/transmission data buffer for wait (exclusively for transmission; data length of 1. Character length of 7, 8, and 9 bits supported)<br>• UART buffer (exclusively for transmission; variable data length from 1 to 9. Character length of 7 and 8 bits supported)<br>• Reception data buffer (exclusively for reception; data length of 1. Character length of 7, 8, and 9 bits supported) | |
| | Data format | Character length: 7 or 8 bits<br>Length of 9 bits supported by using the expansion bit. | |
| | | Transmission stop bit: 1 or 2 bits | |
| | | Parity function: odd, even, 0, or none | |
| | | LSB- or MSB-first transfer selectable | |
| | | Reverse input/output of transmission/reception data possible | |
| | Status | • Transmission status<br>• Reception status<br>• Successful UART buffer transmission<br>• Error detection<br>• Expansion bit detection<br>• ID match<br>• Reset mode status | |
| | Error status | • Bit error<br>• Framing error<br>• Parity error<br>• Overrun error | |
| | Baud rate selection | With the built-in baud rate generator, any baud rate can be set. | |
| | When a certain expansion bit is at the expected level, the data received can be compared to the 8-bit data preset in the register. | | |

Table 19.10 LIN/UART Interface Specifications (3/3)

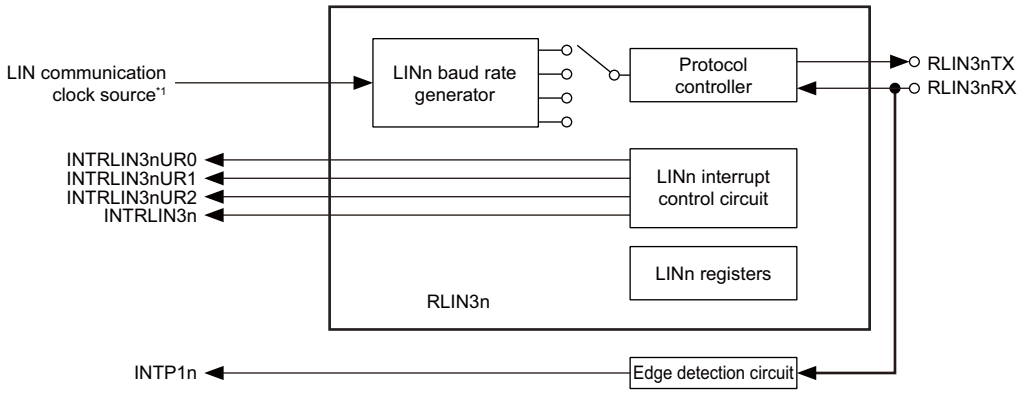
| Item | | Specifications |
|---|---|---|
| UART communication function | Reception of the stop bit is guaranteed. (Start of transmission can be delayed when start of transmission is attempted during reception of the stop bit). | |
| | Interrupt function | • Transmission start/complete<br>• Reception complete<br>• Status/error detection |

Note 1. Since the same register is used for setting, the inter-byte space (header) = response space.

Note 2. For wake-up reception, the input signal low-level width count is indicated.

### 19.2.2 Block Diagram

**Figure 19.1** shows a block diagram of the LIN/UART interface.




Note 1. For the LIN communication clock source, see **Section 12.5.3, Clock Domain Settings**.

**Figure 19.1 LIN/UART Interface Block Diagram**

### 19.2.3 Terms used in block diagram

- RLIN3nTX, RLIN3nRX: LIN/UART interface I/O pins
- LINn baud rate generator: Generates the LIN/UART interface communication clock
- LINn registers: LIN/UART interface registers
- LINn interrupt control circuit: Controls interrupt requests generated by the LIN/UART interface

# 19.3 Registers

## 19.3.1 List of Registers

RLIN3 registers are listed in the following table.

For details about <RLIN3n_base>, see **Section 19.1.2, Register Base Address**.

**Table 19.11 List of Registers (1/2)**

| Module Name | Register Name | Symbol | Address | LIN Master | LIN Slave | UART |
|---|---|---|---|---|---|---|
| RLN3n | LIN wake-up baud rate select register | RLN3nLWBR | <RLIN3n_base> + $01_H$ | √ | √ | √ |
| | LIN / UART baud rate prescaler 01 register | RLN3nLBRP01 | <RLIN3n_base> + $02_H$ | — | √ | √ |
| | LIN / UART baud rate prescaler 0 register | RLN3nLBRP0 | <RLIN3n_base> + $02_H$ | √ | √ | √ |
| | LIN / UART baud rate prescaler 1 register | RLN3nLBRP1 | <RLIN3n_base> + $03_H$ | √ | √ | √ |
| | LIN self-test control register | RLN3nLSTC | <RLIN3n_base> + $04_H$ | √ | √ | — |
| | LIN / UART mode register | RLN3nLMD | <RLIN3n_base> + $08_H$ | √ | √ | √ |
| | LIN break field configuration register/ UART configuration register | RLN3nLBFC | <RLIN3n_base> + $09_H$ | √ | √ | √ |
| | LIN / UART space configuration register | RLN3nLSC | <RLIN3n_base> + $0A_H$ | √ | √ | √ |
| | LIN wake-up configuration register | RLN3nLWUP | <RLIN3n_base> + $0B_H$ | √ | √ | — |
| | LIN interrupt enable register | RLN3nLIE | <RLIN3n_base> + $0C_H$ | √ | √ | — |
| | LIN / UART error detection enable register | RLN3nLEDE | <RLIN3n_base> + $0D_H$ | √ | √ | √ |
| | LIN / UART control register | RLN3nLCUC | <RLIN3n_base> + $0E_H$ | √ | √ | √ |
| | LIN / UART transmission control register | RLN3nLTRC | <RLIN3n_base> + $10_H$ | √ | √ | √ |
| | LIN / UART mode status register | RLN3nLMST | <RLIN3n_base> + $11_H$ | √ | √ | √ |
| | LIN / UART status register | RLN3nLST | <RLIN3n_base> + $12_H$ | √ | √ | √ |
| | LIN / UART error status register | RLN3nLEST | <RLIN3n_base> + $13_H$ | √ | √ | √ |
| | LIN data field configuration register | RLN3nLDFC | <RLIN3n_base> + $14_H$ | √ | √ | √ |
| | LIN / UART ID buffer register | RLN3nLIDB | <RLIN3n_base> + $15_H$ | √ | √ | √ |
| | LIN checksum buffer register | RLN3nLCBR | <RLIN3n_base> + $16_H$ | √ | √ | — |
| | UART data buffer 0 register | RLN3nLUDB0 | <RLIN3n_base> + $17_H$ | — | — | √ |
| | LIN / UART data buffer 1 register | RLN3nLDBR1 | <RLIN3n_base> + $18_H$ | √ | √ | √ |
| | LIN / UART data buffer 2 register | RLN3nLDBR2 | <RLIN3n_base> + $19_H$ | √ | √ | √ |
| | LIN / UART data buffer 3 register | RLN3nLDBR3 | <RLIN3n_base> + $1A_H$ | √ | √ | √ |
| | LIN / UART data buffer 4 register | RLN3nLDBR4 | <RLIN3n_base> + $1B_H$ | √ | √ | √ |
| | LIN / UART data buffer 5 register | RLN3nLDBR5 | <RLIN3n_base> + $1C_H$ | √ | √ | √ |
| | LIN / UART data buffer 6 register | RLN3nLDBR6 | <RLIN3n_base> + $1D_H$ | √ | √ | √ |
| | LIN / UART data buffer 7 register | RLN3nLDBR7 | <RLIN3n_base> + $1E_H$ | √ | √ | √ |
| | LIN / UART data buffer 8 register | RLN3nLDBR8 | <RLIN3n_base> + $1F_H$ | √ | √ | √ |
| | UART operation enable register | RLN3nLUOER | <RLIN3n_base> + $20_H$ | — | — | √ |
| | UART option register 1 | RLN3nLUOR1 | <RLIN3n_base> + $21_H$ | — | — | √ |
| | UART transmission data register | RLN3nLUTDR | <RLIN3n_base> + $24_H$ | — | — | √ |
| | UART transmission data register L | RLN3nLUTDRL | <RLIN3n_base> + $24_H$ | — | — | √ |
| | UART transmission data register H | RLN3nLUTDRH | <RLIN3n_base> + $25_H$ | — | — | √ |
| | UART reception data register | RLN3nLURDR | <RLIN3n_base> + $26_H$ | — | — | √ |
| | UART reception data register L | RLN3nLURDRL | <RLIN3n_base> + $26_H$ | — | — | √ |

**Table 19.11 List of Registers (2/2)**

| Module Name | Register Name | Symbol | Address | LIN Master | LIN Slave | UART |
|---|---|---|---|---|---|---|
| RLN3n | UART reception data register H | RLN3nLURDRH | <RLIN3n_base> + $27_H$ | — | — | √ |
| | UART wait transmission data register | RLN3nLUWTDR | <RLIN3n_base> + $28_H$ | — | — | √ |
| | UART wait transmission data register L | RLN3nLUWTDRL | <RLIN3n_base> + $28_H$ | — | — | √ |
| | UART wait transmission data register H | RLN3nLUWTDRH | <RLIN3n_base> + $29_H$ | — | — | √ |

**Note:** √: Used, —: Not used
When writing to an unused register, write the value after reset.

## 19.3.2 LIN Master Related Registers

### 19.3.2.1 RLN3nLWBR — LIN Wake-Up Baud Rate Select Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $01_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NSPB[3:0] | | | | LPRS[2:0] | | | LWBR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.12 RLN3nLWBR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | NSPB[3:0] | Bit Sampling Count Select<br>b7 b4<br>0 0 0 0: 16 samplings<br>1 1 1 1: 16 samplings<br>Settings other than the above are prohibited. |
| 3 to 1 | LPRS[2:0] | Prescaler Clock Select<br>b3 b1<br>0 0 0: 1/1<br>0 0 1: 1/2<br>0 1 0: 1/4<br>0 1 1: 1/8<br>1 0 0: 1/16<br>1 0 1: 1/32<br>1 1 0: 1/64<br>1 1 1: 1/128 |
| 0 | LWBR0 | Wake-up Baud Rate Select<br>0: In LIN wake-up mode, the clock specified in the LCKS bit of the RLN3nLMD register is used. (LIN1.3)<br>1: In LIN wake-up mode, the clock fa is used regardless of the setting in the LCKS bit of the RLN3nLMD register. (LIN2.x) |

Configure the RLN3nLWBR register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

#### NSPB[3:0] Bits (Bit Sampling Count Select)

These bits select the number of sampling in one Tbit (reciprocal of the baud rate).
In LIN master mode (LIN/UART mode select bits in LIN mode register = $00_B$), set these bits to $0000_B$ or $1111_B$ (16 sampling).

#### LPRS[2:0] Bits (Prescaler Clock Select)

These bits select the frequency division ratio for the prescaler.
The LIN communication clock source is divided by this prescaler.

#### LWBR0 Bit (Wake-up Baud Rate Select)

When LIN Specification Package Revision 1.3 is used, set the LWBR0 bit in the RLN3nLWBR register to 0. This allows an input signal with a low-level width of 2.5 Tbits or more to be measured.
When LIN Specification Package Revision 2.x is used, set the LWBR0 bit to 1. Setting the LWBR0 bit to 1 selects fa as the LIN system clock ($f_{LIN}$) during LIN wake-up mode regardless of the setting of the

RLN3nLMD.LCKS bit (the LCKS bit is not changed). This allows an input signal with a low-level width of 2.5 Tbits or more to be measured.

Setting the baud rate to 19200 bps while fa is selected allows an input signal with a low-level width of 130 us or longer to be detected in the LIN wake-up mode regardless of the setting of the RLN3nLMD.LCKS bit.

### 19.3.2.2 RLN3nLBRP0 — LIN Baud Rate Prescaler 0 Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $02_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LBRP0[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.13 RLN3nLBRP0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | LBRP0[7:0] | Assuming that the value set in this register is N (0 to 255), the baud rate prescaler divides the frequency of the prescaler clock by N + 1.<br>Setting range: $00_H$ to $FF_H$ |

Configure the RLN3nLBRP0 register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

The value set in this register is used to control the frequency of baud rate clock sources fa, fb, and fc.

Assuming that the value set in this register is N, baud rate prescaler 0 divides the frequency of the clock that is selected by the LPRS bits (prescaler clock select bits) by N + 1.

### 19.3.2.3 RLN3nLBRP1 — LIN Baud Rate Prescaler 1 Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $03_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LBRP1[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.14 RLN3nLBRP1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | LBRP1[7:0] | Assuming that the value set in this register is M (0 to 255), the baud rate prescaler divides the frequency of the prescaler clock by M + 1.<br>Setting range: $00_H$ to $FF_H$ |

Configure the RLN3nLBRP1 register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

The value set in this register is used to control the frequency of baud rate clock source fd.

Assuming that the value set in this register is M, baud rate prescaler 1 divides the frequency of the clock that is selected by the LPRS bits (prescaler clock select bits) by M+1.

### 19.3.2.4 RLN3nLSTC — LIN Self-Test Control Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $04_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | LSTM |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.15 RLN3nLSTC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | — | Writing $A7_H$, $58_H$, and $01_H$ successively to the RLN3nLSTC register places the module into LIN self-test mode. |
| 0 | LSTM | LIN Self-Test Mode<br>0: The module is not in LIN self-test mode.<br>1: The module is in LIN self-test mode. |

The RLN3nLSTC register cancels protection of LIN self-test mode.

Configure the RLN3nLSTC register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

Writing $A7_H$, $58_H$, and $01_H$ successively to the RLN3nLSTC register places the module into LIN self-test mode.

When successive writing is completed and the mode is changed to LIN self-test mode, the LSTM bit is set to 1.

Do not write any other value during successive writing.

For making transition to LIN self-test mode, see **Section 19.9, LIN Self-Test Mode**.

When read, bits 6 to 1 return "$000000_B$", and bit 7 returns an undefined value.

#### LSTM Bit (LIN Self-Test Mode)

When transition to LIN self-test mode is completed, the LSTM bit is set to 1.
For exiting LIN self-test mode, see **Section 19.9, LIN Self-Test Mode**.
Writing 1 to this bit does not affect the value of the RLN3nLSTC register if it is not a part of successive writing of $A7_H$, $58_H$, and $01_H$.

### 19.3.2.5 RLN3nLMD — LIN Mode Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base>+ $08_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | LRDNFS | LIOS | LCKS[1:0] | | LMD[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.16 RLN3nLMD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5 | LRDNFS | LIN Reception Data Noise Filter Disable<br>0: The noise filter is enabled.<br>1: The noise filter is disabled. |
| 4 | LIOS | LIN Interrupt Output Select<br>0: RLIN3n interrupt is used.<br>1: RLIN3n transmission interrupt, RLIN3n successful reception interrupt, and RLIN3n status interrupt are used. |
| 3, 2 | LCKS[1:0] | LIN System Clock Select<br>b3 b2<br>0 0: fa (Clock generated by baud rate prescaler 0)<br>0 1: fb (1/2 clock generated by baud rate prescaler 0)<br>1 0: fc (1/8 clock generated by baud rate prescaler 0)<br>1 1: fd (1/2 clock generated by baud rate prescaler 1) |
| 1, 0 | LMD[1:0] | LIN/UART Mode Select<br>b1 b0<br>0 0: LIN master mode |

Configure the RLN3nLMD register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

#### LRDNFS Bit (LIN Reception Data Noise Filter Disable)

The LRDNFS bit enables or disables the noise filter when receiving data.
With 0 set, the noise filter is enabled when receiving data.
With 1 set, the noise filter is disabled when receiving data.

#### LIOS Bit (LIN Interrupt Output Select)

The LIOS bit selects the number of interrupt outputs from the LIN/UART interface.
With 0 set, the RLIN3 interrupt is generated from the LIN/UART interface.
With 1 set, the RLIN3n transmission interrupt, RLIN3n successful reception interrupt, and RLIN3n status interrupt are generated from the LIN/UART interface.
For each interrupt source, see **Section 19.4, Interrupt Sources**.

#### LCKS[1:0] Bits (LIN System Clock Select)

The LCKS bits select the clock to be input to the protocol controller.
With $00_B$ set, the protocol controller is provided with fa (clock generated by baud rate prescaler 0).
With $01_B$ set, the protocol controller is provided with fb (1/2 clock generated by baud rate prescaler 0).
With $10_B$ set, the protocol controller is provided with fc (1/8 clock generated by baud rate prescaler 0).
With $11_B$ set, the protocol controller is provided with fd (1/2 clock generated by baud rate prescaler 1).

When $1_B$ is set in the LWBR0 bit in the RLN3nLWBR register (LIN 2.x), and the RLN3nLMST register is $01_H$ (LIN wake-up mode), the protocol controller is supplied with fa regardless of the setting of this bit (the LCKS bit is not changed).

#### LMD[1:0] Bits (LIN/UART Mode Select)

The LMD bits select the LIN/UART interface mode.
To use the LIN/UART interface as an LIN master, set these bits to $00_B$.

### 19.3.2.6 RLN3nLBFC — LIN Break Field Configuration Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $09_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | BDT[1:0] | | BLT[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.17 RLN3nLBFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5, 4 | BDT[1:0] | Transmission Break Delimiter (High Level) Width Select<br>b5 b4<br>0 0: 1 Tbit<br>0 1: 2 Tbits<br>1 0: 3 Tbits<br>1 1: 4 Tbits |
| 3 to 0 | BLT[3:0] | Transmission Break (Low Level) Width Select<br>b3 b0<br>0 0 0 0: 13 Tbits<br>0 0 0 1: 14 Tbits<br>0 0 1 0: 15 Tbits<br>:<br>1 1 1 0: 27 Tbits<br>1 1 1 1: 28 Tbits |

Set the RLN3nLBFC register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

Some combinations of the specified values result in a frame length exceeding the timeout time. Set the appropriate values in this register.

#### BDT[1:0] Bits (Transmission Break Delimiter (High Level) Width Select)

The BDT bits set the break delimiter (high level) width of transmission frame header.
1 Tbit to 4 Tbits can be set.

#### BLT[3:0] Bits (Transmission Break (Low Level) Width Select)

The BLT bits set the break (low level) width of transmission frame header.
13 Tbits to 28 Tbits can be set.

### 19.3.2.7 RLN3nLSC — LIN Space Configuration Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base>+ $0A_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | IBS[1:0] | | — | IBHS[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R | R/W | R/W | R/W |

**Table 19.18 RLN3nLSC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5, 4 | IBS[1:0] | Inter-Byte Space Select<br>b5 b4<br>0 0: 0 Tbit<br>0 1: 1 Tbit<br>1 0: 2 Tbits<br>1 1: 3 Tbits |
| 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 to 0 | IBHS[2:0] | Inter-Byte Space (Header)/Response Space Select<br>b2 b0<br>0 0 0: 0 Tbit<br>0 0 1: 1 Tbit<br>0 1 0: 2 Tbits<br>0 1 1: 3 Tbits<br>1 0 0: 4 Tbits<br>1 0 1: 5 Tbits<br>1 1 0: 6 Tbits<br>1 1 1: 7 Tbits |

Configure the RLN3nLSC register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

Some combinations of the set values result in the length of a frame or a response exceeding the timeout time. Specify the appropriate values in this register.

#### IBS[1:0] Bits (Inter-Byte Space Select)

The IBS bits set the width of the inter-byte space of the transmission frame response field.
0 Tbit to 3 Tbits can be set.
These bits are enabled only during response transmission; these are disabled during response reception.

#### IBHS[2:0] Bits (Inter-Byte Space (Header)/Response Space Select)

The IBHS bits set the width of the inter-byte space (header) of the transmission frame header field and the response space.
0 Tbit to 7 Tbits can be set.
The response space setting is enabled only during response transmission; setting is disabled during response reception.
The inter-byte space (header) is equal to the response space.

### 19.3.2.8 RLN3nLWUP — LIN Wake-Up Configuration Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base>+ $0B_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WUTL[3:0] | | | | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R | R | R |

**Table 19.19 RLN3nLWUP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | WUTL[3:0] | Wake-up Transmission Low Level Width Select<br>b7 b4<br>0 0 0 0: 1 Tbit<br>0 0 0 1: 2 Tbits<br>0 0 1 0: 3 Tbits<br>0 0 1 1: 4 Tbits<br>:<br>1 1 0 0: 13 Tbits<br>1 1 0 1: 14 Tbits<br>1 1 1 0: 15 Tbits<br>1 1 1 1: 16 Tbits |
| 3 to 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Set the RLN3nLWUP register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

#### WUTL[3:0] Bits (Wake-up Transmission Low Level Width Select)

The WUTL bits set the low level width of the wake-up signal transmission.
1 Tbit to 16 Tbits can be set.

With 1 is set in the LWBR0 bit in the RLN3nLWBR register (LIN 2.x), fa is selected as the LIN system clock ($f_{LIN}$) regardless of the setting of the RLN3nLMD.LCKS bit (the LCKS bit is not changed).

### 19.3.2.9 RLN3nLIE — LIN Interrupt Enable Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $0C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | SHIE | ERRIE | FRCIE | FTCIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 19.20 RLN3nLIE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | SHIE | Successful Header Transmission Interrupt Request Enable<br>0: Disables successful header transmission interrupt request.<br>1: Enables successful header transmission interrupt request. |
| 2 | ERRIE | Error Detection Interrupt Request Enable<br>0: Disables error detection interrupt request.<br>1: Enables error detection interrupt request. |
| 1 | FRCIE | Successful Frame/Wake-up Reception Interrupt Request Enable<br>0: Disables successful frame/wake-up reception interrupt request.<br>1: Enables successful frame/wake-up reception interrupt request. |
| 0 | FTCIE | Successful Frame/Wake-up Transmission Interrupt Request Enable<br>0: Disables successful frame/wake-up transmission interrupt request.<br>1: Enables successful frame/wake-up transmission interrupt request. |

Configure the RLN3nLIE register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

#### SHIE Bit (Successful Header Transmission Interrupt Request Enable)

The SHIE bit enables or disables interrupt request upon successful transmission of a header.
With 0 set, the interrupt request for RLIN3n transmission is not generated when the HTRC flag in the RLN3nLST register is set to 1.
With 1 set, the interrupt request for RLIN3n transmission is generated when the HTRC flag in the RLN3nLST register is set to 1.

#### ERRIE Bit (Error Detection Interrupt Request Enable)

The ERRIE bit enables or disables an interrupt request upon detection of an error.
With 0 set, the interrupt request for RLIN3n status is not generated when the ERR flag in the RLN3nLST register is set to 1.
With 1 set, the interrupt request for RLIN3n status is generated when the ERR flag in the RLN3nLST register is set to 1.
Occurrence factors are bit errors, physical bus errors, frame/response timeout errors, framing errors, checksum errors, and response preparation errors. Detection of the bit error, physical bus error, frame/response timeout error, and framing error can be enabled or disabled using the RLN3nLEDE register.

#### FRCIE Bit (Successful Frame/Wake-up Reception Interrupt Request Enable)

The FRCIE bit enables or disables an interrupt request upon successful reception of a frame or a wake-up signal (input signal low-level width count).

With 0 set, the interrupt request for RLIN3n successful reception is not generated when the FRC flag in the RLN3nLST register is set to 1.
With 1 set, the interrupt request for successful RLIN3n reception is generated when the FRC flag in the RLN3nLST register is set to 1.

#### FTCIE Bit (Successful Frame/Wake-up Transmission Interrupt Request Enable)

The FTCIE bit enables or disables an interrupt request upon successful transmission of a frame or a wake-up signal.
With 0 set, the interrupt request for RLIN3n transmission is not generated when the FTC flag in the RLN3nLST register is set to 1.
With 1 set, the interrupt request for RLIN3n transmission is generated when the FTC flag in the RLN3nLST register is set to 1.

### 19.3.2.10 RLN3nLEDE —LIN Error Detection Enable Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $0D_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LTES | — | — | — | FERE | FTERE | PBERE | BERE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R/W | R/W | R/W | R/W |

**Table 19.21 RLN3nLEDE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | LTES | Timeout Error Select<br>0: Frame timeout error<br>1: Response timeout error |
| 6 to 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | FERE | Framing Error Detection Enable *1<br>0: Disables framing error detection.<br>1: Enables framing error detection. |
| 2 | FTERE | Timeout Error Detection Enable<br>0: Disables frame/response timeout error detection.<br>1: Enables frame/response timeout error detection. |
| 1 | PBERE | Physical Bus Error Detection Enable<br>0: Disables physical bus error detection.<br>1: Enables physical bus error detection. |
| 0 | BERE | Bit Error Detection Enable *1<br>0: Disables bit error detection.<br>1: Enables bit error detection. |

Note 1. Set FERE bit and BERE bit to 1.

Configure the RLN3nLEDE register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

#### LTES Bit (Timeout Error Select)

The LTES bit selects the timeout function to be used.
With 0 set, the timeout function applies to frame timeout.
With 1 set, the timeout function applies to response timeout.
For details on the timeout error, see **Section 19.7.7, Error Statuses**.

#### FERE Bit (Framing Error Detection Enable)

The FERE bit enables or disables detection of the framing error.
With 0 set, the framing error is not detected.
With 1 set, the framing error is detected.
Set this bit to 1. The detection result is indicated in the FER flag in the RLN3nLEST register.
For details on the framing error, see **Section 19.7.7, Error Statuses**.

#### FTERE Bit (Timeout Error Detection Enable)

The FTERE bit enables or disables detection of the frame timeout error or the response timeout error.
With 0 set, the frame timeout error or response timeout error is not detected.
With 1 set, the frame timeout error or response timeout error is detected.

When this bit is set to 1, the detection result is reflected in the FTER flag of the RLN3nLEST register.
With the LTES bit, either the frame timeout error or response timeout error can be selected.
Do not use the timeout error if response data of 9 bytes or more is to be transmitted or received.
For details on the timeout error, see **Section 19.7.7, Error Statuses**.

#### PBERE Bit (Physical Bus Error Detection Enable)

The PBERE bit enables or disables detection of the physical bus error.
With 0 set, the physical bus error is not detected.
With 1 set, the physical bus error is detected.
When this bit is set to 1, the detection result is indicated in the PBER flag in the RLN3nLEST register.
For details on the physical bus error, see **Section 19.7.7, Error Statuses**.

#### BERE Bit (Bit Error Detection Enable)

The BERE bit enables or disables detection of the bit error.
With 0 set, the bit error is not detected.
With 1 set, the bit error is detected.
Set this bit to 1. The detection result is indicated in the BER flag in the RLN3nLEST register.
For details on the bit error, see **Section 19.7.7, Error Statuses**.

### 19.3.2.11 RLN3nLCUC — LIN Control Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $0E_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | OM1 | OM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 19.22 RLN3nLCUC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 1 | OM1 | LIN Mode Select<br>0: LIN wake-up mode<br>1: LIN operation mode |
| 0 | OM0 | LIN Reset<br>0: LIN reset mode<br>1: LIN reset mode is canceled. |

Set the RLN3nLCUC register to $01_H$ to transition to LIN wake-up mode or to $03_H$ to transition to LIN operation mode after exiting LIN reset mode.

In LIN self-test mode, set the RLN3nLCUC register to $03_H$ after a transition to LIN self-test mode is completed.

After a value is written to this register, confirm that the value written is actually indicated in the RLN3nLMST register before writing another value.

#### OM1 Bit (LIN Mode Select)

The OM1 bit selects the specific operation mode (either LIN wake-up mode or LIN operation mode) after canceling LIN reset mode.
With 0 set, LIN wake-up mode.
With 1 set, LIN operation mode.
This bit is enabled only when the OMM0 bit in the RLN3nLMST register is 1.
Writing a value to this bit is disabled while the FTS bit in the RLN3nLTRC register is 1.

#### OM0 Bit (LIN Reset)

The OM0 bit selects either causing a transition to LIN reset mode or canceling LIN reset mode.
With 0 set, LIN reset mode.
With 1 set, LIN reset mode is canceled.

### 19.3.2.12 RLN3nLTRC — LIN Transmission Control Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $10_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | RTS | FTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 19.23 RLN3nLTRC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 1 | RTS | Response Transmission/Reception Start<br>0: Response transmission/reception is stopped in frame separate mode.<br>1: Response transmission/reception is started in frame separate mode. |
| 0 | FTS | Frame Transmission/Wake-up Transmission/Reception Start<br>0: Frame Transmission/wake-up transmission/reception is stopped.<br>1: Frame Transmission/wake-up transmission/reception is started. |

#### RTS Bit (Response Transmission/Reception Start)

In frame separate mode, set the RTS bit to 1 after header transmission is started (FTS bit is 1) and response transmission data is ready. Once set, this bit is automatically cleared to 0 upon completion of frame communication (including error detection) or transition to LIN reset mode.
Only 1 can be written to this bit; 0 cannot be written.
To write 1 to this bit, write $02_H$ to the RLN3nLTRC register using the store instruction.
Writing a value to this bit is disabled when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).
Writing a value to this bit is disabled when the FTS bit is 0 (frame transmission or wake-up transmission/reception is stopped).
When response data of 9 bytes or more is to be transmitted or received, set this bit to 1 each time a data group (variable from 0 to 8 bytes) is transmitted or received. Once set, this bit is automatically cleared to 0 at the end of data group communication or transition to LIN reset mode.

#### FTS Bit (Frame Transmission/Wake-up Transmission/Reception Start)

Set the FTS bit to 1 to start frame transmission and reception.
Also set this bit to 1 to allow wake-up transmission and reception (input signal low-level width count).
Only 1 can be written to this bit; 0 cannot be written.
Writing a value to this bit is disabled when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

This bit is set to 0 upon completion of frame or wake-up communication (including error detection) and transition to LIN reset mode.

### 19.3.2.13 RLN3nLMST — LIN Mode Status Register

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <RLIN3n_base> + $11_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | OMM1 | OMM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 19.24 RLN3nLMST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. |
| 1 | OMM1 | LIN Mode Status Monitor<br>0: LIN wake-up mode.<br>1: LIN operation mode. |
| 0 | OMM0 | LIN Reset Status Monitor<br>0: LIN reset mode.<br>1: Not in LIN reset mode. |

#### OMM1 Bit (LIN Mode Status Monitor)

The OMM1 bit indicates the current operating mode.

#### OMM0 Bit (LIN Reset Status Monitor)

The OMM0 bit indicates the current operating mode.

### 19.3.2.14 RLN3nLST — LIN Status Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $12_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | HTRC | D1RC | — | — | ERR | — | FRC | FTC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R/W | R/W |

**Table 19.25 RLN3nLST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | HTRC | Successful Header Transmission Flag<br>0: Header transmission has not been completed.<br>1: Header transmission has been completed. |
| 6 | D1RC | Successful Data 1 Reception Flag<br>0: Data 1 reception has not been completed.<br>1: Data 1 reception has been completed. |
| 5, 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | ERR | Error Detection Flag<br>0: No error has been detected.<br>1: Error has been detected. |
| 2 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 1 | FRC | Successful Frame/Wake-up Reception Flag<br>0: Frame or wake-up reception has not been completed.<br>1: Frame or wake-up reception has been completed. |
| 0 | FTC | Successful Frame/Wake-up Transmission Flag<br>0: Frame or wake-up transmission has not been completed.<br>1: Frame or wake-up transmission has been completed. |

The RLN3nLST register is automatically cleared to $00_H$ upon transition to LIN reset mode and start of the next communication (when the FTS bit of the RLN3nLTRC register is 1).

In LIN reset mode, this register cannot be written to. In LIN reset mode, the register retains $00_H$.

To clear the specific bits in the register, write 0 to the bits to be cleared and write 1 to the other bits using the store instruction.

#### HTRC Flag (Successful Header Transmission Flag)

Only 0 can be written to the HTRC flag; when 1 is written, the bit retains the value it had before 1 is written.
The HTRC flag is set to 1 upon completion of header transmission. Here, an interrupt request for RLN3n transmission is generated if the SHIE bit in the RLN3nLIE register is 1 (interrupt is enabled).
To clear the bit to 0 before the next communication is started (when the FTS bit of the RLN3nLTRC register is 1), write 0 to the bit while in LIN operation mode.

#### D1RC Flag (Successful Data 1 Reception Flag)

Only 0 can be written to the D1RC flag; when 1 is written, the bit retains the value it had before 1 is written.
The D1RC flag is set to 1 upon completion of data 1 reception. Here, an interrupt request is not generated. To clear the bit to 0 before the next communication is started (when the FTS bit of the

RLN3nLTRC register is 1), write 0 to the bit while in LIN operation mode.
When response data of 9 bytes or more is to be received, this bit is set to 1 each time data 1 of a data group (variable from 0 to 8 bytes) is received. Write 0 before starting reception of the next data group.

### ERR Flag (Error Detection Flag)

The ERR flag is set to 1 upon detection of an error (when at least one of the flags of the RLN3nLEST register is set to 1). Here, an interrupt request for RLN3n status is generated if the ERRIE bit in the RLN3nLIE register is 1 (interrupt is enabled). To clear the bit to 0 before the next communication is started (when the FTS bit of the RLN3nLTRC register is 1), write 0 to the RPER, CSER, FER, FTER, PBER, and BER flags in the RLN3nLEST register while in LIN operation mode or LIN wake-up mode. This clears the ERR flag to 0.

### FRC Flag (Successful Frame/Wake-up Reception Flag)

Only 0 can be written to the FRC flag; when 1 is written, the bit retains the value it had before 1 is written.
The FRC flag is set to 1 upon completion of frame or wake-up reception. Here, an interrupt request for RLN3n successful reception is generated if the FRCIE bit in the RLN3nLIE register is 1 (interrupt is enabled). To clear the bit to 0 before the next communication is started (when the FTS bit of the RLN3nLTRC register is 1), write 0 to the bit while in LIN operation mode or LIN wake-up mode.
When response data of 9 bytes or more is to be received, this bit is set to 1 each time a data group (variable from 0 to 8 bytes) is received. Write 0 before starting reception of the next data group.

### FTC Flag (Successful Frame/Wake-up Transmission Flag)

Only 0 can be written to the FTC flag; when 1 is written, the bit retains the value it had before 1 is written.
The FTC flag is set to 1 upon completion of frame or wake-up transmission. Here, an interrupt request for RLN3n transmission is generated if the FTCIE bit in the RLN3nLIE register is 1 (interrupt is enabled). To clear the bit to 0 before the next communication is started (when the FTS bit of the RLN3nLTRC register is 1), write 0 to the bit while in LIN operation mode or LIN wake-up mode.
When response data of 9 bytes or more is to be transmitted, this bit is set to 1 each time a data group (variable from 0 to 8 bytes) is transmitted. Write 0 before starting transmission of the next data group.

### 19.3.2.15 RLN3nLEST — LIN Error Status Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $13_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RPER | — | CSER | — | FER | FTER | PBER | BER |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R/W | R | R/W | R/W | R/W | R/W |

**Table 19.26 RLN3nLEST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RPER | Response Preparation Error Flag<br>0: Response preparation error has not been detected.<br>1: Response preparation error has been detected. |
| 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5 | CSER | Checksum Error Flag<br>0: Checksum error has not been detected.<br>1: Checksum error has been detected. |
| 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | FER | Framing Error Flag<br>0: Framing error has not been detected.<br>1: Framing error has been detected. |
| 2 | FTER | Timeout Error Flag<br>0: Frame/response timeout error has not been detected.<br>1: Frame/response timeout error has been detected. |
| 1 | PBER | Physical Bus Error Flag<br>0: Physical bus error has not been detected.<br>1: Physical bus error has been detected. |
| 0 | BER | Bit Error Flag<br>0: Bit error has not been detected.<br>1: Bit error has been detected. |

The RLN3nLEST register is automatically cleared to $00_H$ upon transition to LIN reset mode and start of the next communication (when the FTS bit of the RLN3nLTRC register is 1).

In LIN reset mode, this register cannot be written to. In LIN reset mode, the register retains $00_H$.

When the FTS bit in the RLN3nLTRC register is 1 (frame transmission or wake-up transmission/ reception is started), do not write a value to this register.

To clear the specific bits in the register, write 0 to the bits to be cleared and write 1 to the other bits using the store instruction.

#### RPER Flag (Response Preparation Error Flag)

Only 0 can be written to the RPER flag; when 1 is written, the bit retains the value it had before 1 is written.

The RPER flag is set to 1 upon response preparation error detection. To clear the bit to 0 before the next communication is started (when the FTS bit of the RLN3nLTRC register is 1), write 0 to the bit while in LIN operation mode.

#### CSER Flag (Checksum Error Flag)

Only 0 can be written to the CSER flag; when 1 is written, the bit retains the value it had before 1 is written.
The CSER flag is set to 1 upon checksum error detection. To clear the bit to 0 before the next communication is started (when the FTS bit of the RLN3nLTRC register is 1), write 0 to the bit while in LIN operation mode.

#### FER Flag (Framing Error Flag)

Only 0 can be written to the FER flag; when 1 is written, the bit retains the value it had before 1 is written.
When the value of the FERE bit of the RLN3nLEDE register is 1 (framing error detection enabled), the FER flag is set to 1 upon framing error detection. To clear the bit to 0 before the next communication is started (when the FTS bit of the RLN3nLTRC register is 1), write 0 to the bit while in LIN operation mode.

#### FTER Flag (Timeout Error Flag)

Only 0 can be written to the FTER flag; when 1 is written, the bit retains the value it had before 1 is written.
When the FTERE bit of the RLN3nLEDE register is 1 (frame/response timeout error detection enabled), the FTER flag is set to 1 upon frame timeout error or response timeout error detection. To clear the bit to 0 before the next communication is started (when the FTS bit of the RLN3nLTRC register is 1), write 0 to the bit while in LIN operation mode.

#### PBER Flag (Physical Bus Error Flag)

Only 0 can be written to the PBER flag; when 1 is written, the bit retains the value it had before 1 is written.
When the PBERE bit of the RLN3nLEDE register is 1 (physical bus error detection enabled), the PBER flag is set to 1 upon physical bus error detection. To clear the bit to 0 before the next communication is started (when the FTS bit of the RLN3nLTRC register is 1), write 0 to the bit while in LIN operation mode or LIN wake-up mode.

#### BER Flag (Bit Error Flag)

Only 0 can be written to the BER flag; when 1 is written, the bit retains the value it had before 1 is written.
When the BERE bit of the RLN3nLEDE register is 1 (bit error detection enabled), the BER flag is set to 1 upon bit error detection. To clear the bit to 0 before the next communication is started (when the FTS bit of the RLN3nLTRC register is 1), write 0 to the bit while in LIN operation mode or LIN wake-up mode.

### 19.3.2.16 RLN3nLDFC — LIN Data Field Configuration Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $14_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LSS | FSM | CSM | RFT | RFDL[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.27 RLN3nLDFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | LSS | Transmission/Reception Continuation Select<br>0: The data group to be transmitted/received next is the last one.<br>1: The data group to be transmitted/received next is not the last one. (Checksum is not included.) |
| 6 | FSM | Frame Separate Mode Select<br>0: Frame separate mode is not set.<br>1: Frame separate mode is set. |
| 5 | CSM | Checksum Select<br>0: Classic checksum mode<br>1: Enhanced checksum mode |
| 4 | RFT | Response Field Communication Direction Select<br>0: Reception<br>1: Transmission |
| 3 to 0 | RFDL[3:0] | Response Field Length Select<br>b3 b0<br>0 0 0 0: 0 byte (+ checksum)<br>0 0 0 1: 1 byte (+ checksum)<br>0 0 1 0: 2 bytes (+ checksum)<br>:<br>0 1 1 1: 7 bytes (+ checksum)<br>1 0 0 0: 8 bytes (+ checksum)<br>Settings other than the above are prohibited. |

#### LSS Bit (Transmission/Reception Continuation Select)

The LSS bit indicates that the data group to be transmitted or received next is not the last data group when response data of 9 bytes or more is to be transmitted or received.
With 0 set, data and checksum are transmitted or received because the next data group to be transmitted or received is the last one.
With 1 set, only data is transmitted or received, and the checksum is not included because the next data group to be transmitted or received is not the last one.
Set the LSS bit only when the FSM bit is 1 (frame separate mode) and response data of 9 bytes or more is to be transmitted or received.

Set the LSS bit only when the RTS bit in the RLN3nLTRC is 0 (response transmit/receive is stopped).

#### FSM Bit (Frame Separate Mode Select)

The FSM bit selects the response communication mode.
With 0 set, frame separate mode is not selected. In this case, after header transmission is started (the FTS bit in the RLN3nLTRC register is 1), response is transmitted/received without setting the RTS bit in the RLN3nLTRC register.

With 1 set, frame separate mode is selected. If the RTS bit of the RLN3nLTRC register is set to 1 during header transmission, response transmission is executed after header transmission is completed.

For response reception which is 8 bytes or less (the RFT bit is 0), set the FSM bit to 0.
When transitioning to LIN self-test mode, set this bit to 0 before transition.
For details on frame separate mode, see **Section 19.7.4.1, Transmission of LIN Frames**.
Set this bit when the FTS bit in the RLN3nLTRC register is 0 (frame transmission or wake-up transmission/reception is stopped).
When response data of 9 bytes or more is to be transmitted or received, set the FSM bit to 1.

#### CSM Bit (Checksum Select)

The CSM bit selects the checksum mode.
With 0 set, classic checksum mode is selected.
With 1 set, enhanced checksum mode is selected.
When the timeout error detection is enabled (the FTERE bit in the RLN3nLEDE register is 1), the specific timeout time depends on the setting of this bit. For details, see **Section 19.7.7, Error Statuses**.
Set this bit when the FTS bit in the RLN3nLTRC register is 0 (frame transmission or wake-up transmission/reception is stopped).
When response data of 9 bytes or more is to be transmitted or received, do not change the CSM bit setting after the first data group through the last data group.
During communication of response data of 9 bytes or more, only the last data group (the LSS bit is 0) includes the checksum, and no other groups (the LSS bit is 1) include the checksum.

#### RFT Bit (Response Field Communication Direction Select)

The RFT bit sets the direction of the response field/wake-up signal communication.
With 0 set, reception is performed in the response field. In LIN wake-up mode, wake-up reception is performed (input signal low-level width count).
With 1 set, transmission is performed in the response field. In LIN wake-up mode, wake-up transmission is performed.
Set this bit when the FTS bit in the RLN3nLTRC register is 0 (frame transmission or wake-up transmission/reception is stopped).
When response data of 9 bytes or more is to be transmitted or received, do not change the RFT bit setting after the first data group through the last data group.

#### RFDL[3:0] Bits (Response Field Length Select)

The RFDL bits set the length of the response field data.
The data length can be 0 to 8 bytes excluding the checksum size.
To transmit response data with the FSM bit set to 0 (not frame separate mode), set the RFDL bits before header transmission (the FTS bit in the RLN3nLTRC register is 0).
To transmit response data with the FSM bit set to 1 (frame separate mode), set the RFDL bits before response transmission (the RTS bit in the RLN3nLTRC register is 0).
To receive response data, set the RFDL bits before header transmission (the FTS bit in the RLN3nLTRC register is 0).
When response data of 9 bytes or more is to be transmitted or received, set the RFDL bits before data group transmission/reception (RTS bit in the RLN3nLTRC register is 0).
Only the last data group (the LSS bit is 0) includes the checksum, and no other groups (the LSS bit is 1) include the checksum.

### 19.3.2.17 RLN3nLIDB — LIN ID Buffer Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $15_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IDP1 | IDP0 | ID[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.28 RLN3nLIDB Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | IDP1 | Parity Setting (P1)<br>Sets the parity bits (P1) to be transmitted in the ID field. |
| 6 | IDP0 | Parity Setting (P0)<br>Sets the parity bits (P0) to be transmitted in the ID field. |
| 5 to 0 | ID[5:0] | ID Setting<br>Sets the 6-bit ID value to be transmitted in the ID field. |

Set the RLN3nLIDB register when the FTS bit in the RLN3nLTRC register is 0 (frame transmission or wake-up transmission/reception is stopped).

In LIN self-test mode, this register operates as described below.

Write the value to be transmitted before communication. After completion of frame transmission/ reception (after loopback), the reversed value of the received value can be read.

For details about the LIN self-test mode, see **Section 19.9, LIN Self-Test Mode**.

#### IDP[1:0] Bits (Parity Setting)

The IDP bits set the parity bits (P0 and P1) to be transmitted in the ID field of the LIN frame. IDP0 is for P0 and IDP1 is for P1.

Since parity is not automatically calculated, set the calculation result. Note that if the erroneous result is set, it is transmitted as is.

#### ID[5:0] Bits (ID Setting)

The ID bits set the 6-bit ID value to be transmitted in the ID field of the LIN frame.

### 19.3.2.18 RLN3nLCBR — LIN Checksum Buffer Register

**Access:** This register is a read-only register that can be read in 8-bit units. In LIN self-test mode, this register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $16_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CKSM[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.29 RLN3nLCBR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | CKSM[7:0] | Stores the checksum value transmitted or received. |

In LIN mode, this register operates as follows:

- When the RFT bit in the RLN3nLDFC register is 1 (transmission):
  The value transmitted can be read from the register. Read the value after transmission is completed.
  Writing to this register is invalid.
- When the RFT bit in the RLN3nLDFC register is 0 (reception):
  The value received can be read from the register. Read the value after reception is completed.
  Writing to this register is invalid.

When response data of 9 bytes or more is to be transmitted or received, the checksum is appended only to the last data group; this register is not updated for the other data groups.

In LIN self-test mode, this register operates as follows:

- When the RFT bit in the RLN3nLDFC register is 1 (transmission):
  After completion of the frame transmission (after loopback), the reversed value of the received value can be read.
- When the RFT bit in the RLN3nLDFC register is 0 (reception):
  Write the value to be received before communication. After completion of frame transmission/reception (after loopback), the reversed value of the received value can be read.

For details about the LIN self-test mode, see **Section 19.9, LIN Self-Test Mode**.

Set the RLN3nLCBR register when the FTS bit in the RLN3nLTRC register is 0 (frame transmission or wake-up transmission/reception is stopped).

### 19.3.2.19 RLN3nLDBRb — LIN Data Buffer b Register (b = 1 to 8)

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN3nLDBR1: <RLIN3n_base> + $18_H$
RLN3nLDBR2: <RLIN3n_base> + $19_H$
RLN3nLDBR3: <RLIN3n_base> + $1A_H$
RLN3nLDBR4: <RLIN3n_base> + $1B_H$
RLN3nLDBR5: <RLIN3n_base> + $1C_H$
RLN3nLDBR6: <RLIN3n_base> + $1D_H$
RLN3nLDBR7: <RLIN3n_base> + $1E_H$
RLN3nLDBR8: <RLIN3n_base> + $1F_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LDB[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.30 RLN3nLDBRb Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | LDB[7:0] | Sets the data to be transmitted or reads the received data.<br>Setting range: $00_H$ to $FF_H$ |

- For response transmission:
  These registers set the data to be transmitted in the response field.
  Use these registers with the following settings.
  - RFT in RLN3nLDFC register is 1 (transmission)
  - FSM in RLN3nLDFC register is 0 (not frame separate mode)
  - FTS in RLN3nLTRC register is 0 (frame transmission or wake-up transmission/reception is stopped)

  or
  - RFT in RLN3nLDFC register is 1 (transmission)
  - FSM in RLN3nLDFC register is 1 (frame separate mode)
  - RTS in RLN3nLTRC register is 0 (response transmission/reception is stopped)
- For response reception:
  These registers store the data received in the response field.
  The received data is overwritten. If an error is detected, the data prior to reception interruption is stored in the register.
  Do not read these registers when the FTS bit is 1 (frame transmission or wake-up transmission/reception is started)
- For transmission of response data of 9 bytes or more:
  Use these registers with the following settings.
  - RFT in RLN3nLDFC register is 1 (transmission)
  - FSM in RLN3nLDFC register is 1 (frame separate mode)
  - RTS in RLN3nLTRC register is 0 (response transmission/reception is stopped)
- For reception of response data of 9 bytes or more:
  Do not read these registers when the RTS bit is 1 (response transmission/reception is started).

In LIN self-test mode, these registers operate as described below.

Write the value to be transmitted before communication. After completion of frame transmission/ reception (after loopback), the reversed value of the received value can be read.

For details about the LIN self-test mode, see **Section 19.9, LIN Self-Test Mode**.

## 19.3.3 LIN Slave Related Registers

### 19.3.3.1 RLN3nLWBR — LIN Wake-Up Baud Rate Select Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $01_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NSPB[3:0] | | | | LPRS[2:0] | | | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R |

**Table 19.31 RLN3nLWBR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | NSPB[3:0] | Bit Sampling Count Select<br>b7 b4<br>0 0 0 0: 16 samplings<br>0 0 1 1: 4 samplings<br>0 1 1 1: 8 samplings<br>1 1 1 1: 16 samplings<br>Settings other than the above are prohibited. |
| 3 to 1 | LPRS[2:0] | Prescaler Clock Select<br>b3 b1<br>0 0 0: 1/1<br>0 0 1: 1/2<br>0 1 0: 1/4<br>0 1 1: 1/8<br>1 0 0: 1/16<br>1 0 1: 1/32<br>1 1 0: 1/64<br>1 1 1: 1/128 |
| 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Configure the RLN3nLWBR register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

#### NSPB[3:0] Bits (Bit Sampling Count Select)

The NSPB bits select the number of sampling in one Tbit (reciprocal of the baud rate).
When the frame communication is performed in LIN slave mode (fixed baud rate) (LMD[1:0] bits in the RLN3nLMD register = $11_B$), set these bits to "$0000_B$" or "$1111_B$" (16 samplings).
When the frame communication is performed in LIN slave mode (auto baud rate) (LMD[1:0] bits in the RLN3nLMD register = $10_B$), set these bits to "$0011_B$" (4 samplings) or "$0111_B$" (8 samplings).

#### LPRS[2:0] Bits (Prescaler Clock Select)

The LPRS bits select the frequency division ratio for the prescaler. The LIN communication clock source is divided by this prescaler.

In LIN slave mode (auto baud rate) (LMD[1:0] bits in the RLN3nLMD register = $10_B$), set these bits so that the prescaler clock becomes as follows according to the target baud rate.

| [Target baud rate] | [Prescaler clock] |
|---|---|
| 1 kbps to 20 kbps | : 4MHz*1 |
| 1 kbps to 2.4 kbps (excluding 2.4 kbps) | : 4MHz |
| 2.4 kbps to 20 kbps | : 8 MHz to 12 MHz |

**Note 1.** Use the clock with NSPB bits set to "$0011_B$" (4 samplings).

### 19.3.3.2 RLN3nLBRP01 — LIN Baud Rate Prescaler 01 Register

**Access:** RLN3nLBRP01 can be read or written in 16-bit units.
RLN3nLBRP0 can be read or written in 8-bit units.
RLN3nLBRP1 can be read or written in 8-bit units.

**Address:** RLN3nLBRP01: <RLIN3n_base> + $02_H$
RLN3nLBRP0: <RLIN3n_base> + $02_H$
RLN3nLBRP1: <RLIN3n_base> + $03_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BRP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.32 RLN3nLBRP01 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | BRP[15:0] | Assuming that the value set in this register is L (0 to 65535), the baud rate prescaler divides the frequency of the prescaler clock by L + 1.<br>Setting range: $0000_H$ to $FFFF_H$ |

Configure the RLN3nLBRP01 register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

Assuming that the value set in this register is L, the baud rate prescaler divides the frequency of the clock that is selected by the LPRS bits (prescaler clock select bits) in the RLN3nLWBR register by L + 1.

The RLN3nLBRP01 register can be accessed in 8-bit units using registers RLN3nLBRP0 and RLN3nLBRP1.

**NOTE**

In LIN slave mode [auto baud rate], the system automatically sets the result of baud rate correction to the RLN3nLBRP01 register on successful reception of the sync field.

### 19.3.3.3 RLN3nLSTC — LIN Self-Test Control Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $04_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | LSTM |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.33 RLN3nLSTC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | - | Writing $A7_H$, $58_H$, and $01_H$ successively to the RLN3nLSTC register places the module into LIN self-test mode. |
| 0 | LSTM | LIN Self-Test Mode<br>0: The module is not in LIN self-test mode.<br>1: The module is in LIN self-test mode. |

The RLN3nLSTC register cancels protection of LIN self-test mode.

Configure the RLN3nLSTC register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

Writing $A7_H$, $58_H$, and $01_H$ successively to the RLN3nLSTC register places the module into LIN self-test mode.

When successive writing is completed and the module is placed in LIN self-test mode, the LSTM bit is set to 1.

Do not write any other value during successive writing.

For making transition to LIN self-test mode, see **Section 19.9, LIN Self-Test Mode**.

When read, bits 6 to 1 return "$000000_B$", and bit 7 returns an undefined value.

#### LSTM Bit (LIN Self-Test Mode)

When transition to LIN self-test mode is completed, the LSTM bit is set to 1.
For exiting LIN self-test mode, see **Section 19.9, LIN Self-Test Mode**.
Writing 1 to this bit does not affect the value of the RLN3nLSTC register if it is not a part of successive writing of $A7_H$, $58_H$, and $01_H$.

### 19.3.3.4 RLN3nLMD — LIN Mode Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $08_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | LRDNFS | LIOS | — | — | LMD[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R | R | R/W | R/W |

**Table 19.34 RLN3nLMD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5 | LRDNFS | LIN Reception Data Noise Filter Disable<br>0: The noise filter is enabled.<br>1: The noise filter is disabled. |
| 4 | LIOS | LIN Interrupt Output Select<br>0: RLIN3 interrupt is used.<br>1: RLIN3n transmission interrupt, RLIN3n successful reception interrupt, and RLIN3n reception status interrupt are used. |
| 3, 2 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 1, 0 | LMD[1:0] | LIN/UART Mode Select<br>b1 b0<br>1 0: LIN Slave mode (auto baud rate)<br>1 1: LIN Slave mode (fixed baud rate) |

Configure the RLN3nLMD register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

#### LRDNFS Bit (LIN Reception Data Noise Filter Disable)

The LRDNFS bit enables or disables the noise filter when receiving data.
With 0 set, the noise filter is enabled when receiving data.
With 1 set, the noise filter is disabled when receiving data.

#### LIOS Bit (LIN Interrupt Output Select)

The LIOS bit selects the number of interrupt outputs from the LIN/UART interface.
With 0 set, the RLIN3 interrupt is generated from the LIN/UART interface.
With 1 set, the RLIN3n transmission interrupt, RLIN3n successful reception interrupt, and RLIN3n reception status interrupt are generated from the LIN/UART interface.
For each interrupt source, see **Section 19.4, Interrupt Sources**.

#### LMD[1:0] Bits (LIN/UART Mode Select)

The LMD bits select the LIN/UART interface mode.
To use this module as an LIN slave, set these bits to "$10_B$" (auto baud rate) or "$11_B$" (fixed baud rate).

### 19.3.3.5 RLN3nLBFC — LIN Break Field Configuration Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $09_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | LBLT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 19.35 RLN3nLBFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | LBLT | Reception Break (Low-Level) Detection Width Setting<br>0: A break (low-level) is detected in 9.5 or 10 Tbits<br>1: A break (low-level) is detected in 10.5 or 11 Tbits |

Configure the RLN3nLBFC register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

#### LBLT Bit (Reception Break (Low-Level) Detection Width Setting)

- When RLN3nLMD.LMD is "$10_B$" (LIN slave mode (auto baud rate))
  - 0: Low-level width of 10 Tbits or longer is detected.
  - 1: Low-level width of 11 Tbits or longer is detected.
- When RLN3nLMD.LMD is "$11_B$" (LIN slave mode (fixed baud rate))
  - 0: Low-level width of 9.5 Tbits or longer is detected.
  - 1: Low-level width of 10.5 Tbits or longer is detected.

### 19.3.3.6 RLN3nLSC — LIN Space Configuration Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $0A_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | IBS[1:0] | | — | IBHS[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R | R/W | R/W | R/W |

**Table 19.36 RLN3nLSC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5, 4 | IBS[1:0] | Inter-Byte Space Select<br>b5 b4<br>0 0: 0 Tbit<br>0 1: 1 Tbit<br>1 0: 2 Tbits<br>1 1: 3 Tbits |
| 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 to 0 | IBHS[2:0] | Response Space Setting<br>b2 b0<br>0 0 0: 0 Tbit<br>0 0 1: 1 Tbit<br>0 1 0: 2 Tbits<br>0 1 1: 3 Tbits<br>1 0 0: 4 Tbits<br>1 0 1: 5 Tbits<br>1 1 0: 6 Tbits<br>1 1 1: 7 Tbits |

Configure the RLN3nLSC register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

This register is enabled only during response transmission, and disabled during response reception.

Some combinations of the specified values result in a frame or response length exceeding the timeout time. Specify the appropriate values in this register.

#### IBS[1:0] Bits (Inter-Byte Space Select)

The IBS bits set the inter-byte space width of the response transmission.
0 Tbit to 3 Tbits can be set.

#### IBHS[2:0] Bits (Inter-Byte Space (Header)/Response Space Select)

The IBHS bits set the transmission width of the response space.
0 Tbit to 7 Tbits can be set.

### 19.3.3.7 RLN3nLWUP — LIN Wake-Up Configuration Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $0B_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WUTL[3:0] | | | | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R | R | R |

**Table 19.37 RLN3nLWUP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | WUTL[3:0] | Wake-up Transmission Low level Width Select<br>b7 b4<br>0 0 0 0: 1 Tbit<br>0 0 0 1: 2 Tbits<br>0 0 1 0: 3 Tbits<br>0 0 1 1: 4 Tbits<br>:<br>1 1 0 0: 13 Tbits<br>1 1 0 1: 14 Tbits<br>1 1 1 0: 15 Tbits<br>1 1 1 1: 16 Tbits |
| 3 to 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Configure the RLN3nLWUP register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

#### WUTL[3:0] Bits (Wake-up Transmission Low Level Width Select)

The WUTL bits set the low-level width of the wake-up frame transmission.
1 Tbit to 16 Tbits can be set.

### 19.3.3.8 RLN3nLIE — LIN Interrupt Enable Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $0C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | SHIE | ERRIE | FRCIE | FTCIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 19.38 RLN3nLIE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | SHIE | Successful Header Reception Interrupt Request Enable<br>0: Disables successful header reception interrupt request.<br>1: Enables successful header reception interrupt request. |
| 2 | ERRIE | Error Detection Interrupt Request Enable<br>0: Disables error detection interrupt request.<br>1: Enables error detection interrupt request. |
| 1 | FRCIE | Successful Response/Wake-up Reception Interrupt Request Enable<br>0: Disables successful Response/wake-up reception interrupt request.<br>1: Enables successful Response/wake-up reception interrupt request. |
| 0 | FTCIE | Successful Response/Wake-up Transmission Interrupt Request Enable<br>0: Disables successful Response/wake-up transmission interrupt request.<br>1: Enables successful Response/wake-up transmission interrupt request. |

Configure the RLN3nLIE register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

#### SHIE Bit (Successful Header Reception Interrupt Request Enable)

The SHIE bit enables or disables an interrupt request upon successful reception of a header.
With 0 set, the interrupt request for RLIN3n successful reception is not generated when the HTRC flag in the RLN3nLST register is set to 1.
With 1 set, the interrupt request for RLIN3n successful reception is generated when the HTRC flag in the RLN3nLST register is set to 1.

#### ERRIE Bit (Error Detection Interrupt Request Enable)

The ERRIE bit enables or disables an interrupt request upon detection of an error.
With 0 set, the interrupt request for RLIN3n status is not generated when the ERR flag in the RLN3nLST register is set to 1.
With 1 set, the interrupt request for RLIN3n status is generated when the ERR flag in the RLN3nLST register is set to 1.
Error types that are interrupt sources are the bit error, frame/response timeout error, framing error, sync filed error, ID parity error, checksum error, and response preparation error.
Detection of the bit error, frame/response timeout error, sync filed error, ID parity error, and framing error can be enabled or disabled using the RLN3nLEDE register.

### FRCIE Bit (Successful Response/Wake-up Reception Interrupt Request Enable)

The FRCIE bit enables or disables an interrupt request upon successful reception of a response or a wake-up frame (input signal low-level width count).
With 0 set, the interrupt request for RLIN3n successful reception is not generated when the FRC flag in the RLN3nLST register is set to 1.
With 1 set, the interrupt request for RLIN3n successful reception is generated when the FRC flag in the RLN3nLST register is set to 1.

### FTCIE Bit (Successful Response/Wake-up Transmission Interrupt Request Enable)

The FTCIE bit enables or disables an interrupt request upon successful transmission of a response or a wake-up frame.
With 0 set, the interrupt request for RLIN3n successful transmission is not generated when the FTC flag in the RLN3nLST register is set to 1.
With 1 set, the interrupt request for RLIN3n successful transmission is generated when the FTC flag in the RLN3nLST register is set to 1.

### 19.3.3.9 RLN3nLEDE — LIN Error Detection Enable Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $0D_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LTES | IPERE | — | SFERE | FERE | TERE | — | BERE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W |

**Table 19.39 RLN3nLEDE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | LTES | Timeout Error Select<br>0: Frame timeout error<br>1: Response timeout error |
| 6 | IPERE | ID Parity Error Detection Enable<br>0: Disables ID Parity error detection.<br>1: Enables ID Parity error detection. |
| 5 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 4 | SFERE | Sync Field Error Detection Enable<br>0: Disables Sync Field error detection.<br>1: Enables Sync Field error detection. |
| 3 | FERE | Framing Error Detection Enable *1<br>0: Disables framing error detection.<br>1: Enables framing error detection. |
| 2 | TERE | Timeout Error Detection Enable<br>0: Disables frame/response timeout error detection.<br>1: Enables frame/response timeout error detection. |
| 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | BERE | Bit Error Detection Enable *1<br>0: Disables bit error detection.<br>1: Enables bit error detection. |

Note 1. Set FERE bit and BERE bit to 1

Configure the RLN3nLEDE register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

#### LTES Bit (Timeout Error Select)

The LTES bit selects the timeout function to be used.
With 0 set, the timeout function applies to frame timeout.
With 1 set, the timeout function applies to response timeout.
For details on the timeout error, see **Section 19.7.7, Error Statuses**.

#### IPERE Bit (ID Parity Error Detection Enable)

This bit enables or disables detection of the ID parity error.
With 0 set, the ID parity error is not detected.
With 1 set, the ID parity error is detected.
When this bit is set to 1, the detection result is reflected in the IPER flag of the RLN3nLEST register.
For details on the ID parity error, see **Section 19.7.7, Error Statuses**.

#### SFERE Bit (Sync Field Error Detection Enable)

This bit enables or disables detection of the sync field error.
With 0 set, the sync field error is not detected.
With 1 set, the sync field error is detected.
Regardless of the setting of this bit, when a sync field error is detected, this module waits for the next header.
When this bit is set to 1, the detection result is reflected in the SFER flag of the RLN3nLEST register.
For details on the sync filed error, see **Section 19.7.7, Error Statuses**.

#### FERE Bit (Framing Error Detection Enable)

The FERE bit enables or disables detection of the framing error.
With 0 set, the framing error is not detected.
With 1 set, the framing error is detected.
Set this bit to 1. The detection result is indicated in the FER flag in the RLN3nLEST register.
For details on the framing error, see **Section 19.7.7, Error Statuses**.

#### TERE Bit (Timeout Error Detection Enable)

The TERE bit enables or disables detection of the frame timeout error or the response timeout error.
With 0 set, the frame timeout error or response timeout error is not detected.
With 1 set, the frame timeout error or response timeout error is detected.
When this bit is set to 1, the detection result is reflected in the TER flag of the RLN3nLEST register.
With the LTES bit, either the frame timeout error or response timeout error can be selected.
The timeout error should not be used in LIN slave mode [auto baud rate] (when the LMD[1:0] bits in the RLN3nLMD register are "$10_B$").
Do not use the timeout error if response data of 9 bytes or more is to be transmitted or received.
For details on the timeout error, see **Section 19.7.7, Error Statuses**.

#### BERE Bit (Bit Error Detection Enable)

The BERE bit enables or disables detection of the bit error.
With 0 set, the bit error is not detected.
With 1 set, the bit error is detected.
Set this bit to 1. The detection result is indicated in the BER flag in the RLN3nLEST register.
For details on the bit error, see **Section 19.7.7, Error Statuses**.

### 19.3.3.10 RLN3nLCUC — LIN Control Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $0E_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | OM1 | OM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 19.40 RLN3nLCUC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 1 | OM1 | LIN Mode Select<br>0: LIN wake-up mode<br>1: LIN operation mode |
| 0 | OM0 | LIN Reset<br>0: LIN reset mode<br>1: LIN reset mode is canceled. |

Set the RLN3nLCUC register to $01_H$ to transition to LIN wake-up mode or to $03_H$ to transition to LIN operation mode after exiting LIN reset mode.

In LIN self-test mode, set the RLN3nLCUC register to $03_H$ after a transition to LIN self-test mode is completed.

After a value is written to this register, confirm that the value written is actually indicated in the RLN3nLMST register before writing another value.

#### OM1 Bit (LIN Mode Select)

The OM1 bit selects the specific LIN operation mode (LIN wake-up mode or LIN operation mode) that is entered after exiting LIN reset mode
With 0 set, the LIN/UART interface enters LIN wake-up mode.
With 1 set, the LIN/UART interface enters LIN operation mode.
This bit is enabled only when the OMM0 bit in the RLN3nLMST register is 1.
Writing a value to this bit is disabled while the FTS bit in the RLN3nLTRC register is 1.

#### OM0 Bit (LIN Reset)

The OM0 bit selects whether to transition to or exit LIN reset mode.
With 0 set, the LIN/UART interface enters LIN reset mode.
With 1 set, the LIN/UART interface exits LIN reset mode.

### 19.3.3.11 RLN3nLTRC — LIN Transmission Control Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $10_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | LNRR | RTS | FTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W |

**Table 19.41 RLN3nLTRC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 | LNRR | No LIN Response Request<br>0: Response for the reception ID<br>1: No response for the reception ID |
| 1 | RTS | Response Transmission/Reception Start<br>0: Response transmission/reception is stopped.<br>1: Response transmission/reception is started. |
| 0 | FTS | LIN Communication Start<br>0: Header reception/wake-up transmission/reception is stopped.<br>1: Header reception/wake-up transmission/reception is started. |

#### LNRR Bit (No LIN Response Request)

Set this bit to 1 if no response is to be transmitted/received after receiving the header and checking the received ID.

Once set, this bit is automatically cleared to 0 upon detection of new sync field or transition to LIN reset mode.

Only 1 can be written to this bit; 0 cannot be written.

To write 1 to this bit, write $04_H$ using the store instruction.

Do not set this bit and the RTS bit to 1 simultaneously.

Writing a value to this bit is disabled when the OMM0 bit of the RLN3nLMST register is $0_B$ (LIN reset mode).

Writing a value to this bit is disabled when the FTS bit is 0 (header reception or wake-up transmission/reception is stopped).

When response data of 9 bytes or more is to be transmitted or received, use this bit only after the completion of the header. (Do not use this bit for the second or later data group.)

#### RTS Bit (Response Transmission/Reception Start)

Set this bit to 1 to start response transmission or reception after receiving the header and checking the received ID.

Once set, this bit is automatically cleared to 0 upon completion of response transmission or reception (including error detection) or transition to LIN reset mode.

Only 1 can be written to this bit; 0 cannot be written.

To write 1 to this bit, write $02_H$ to the RLN3nLTRC register using the store instruction.

Do not set this bit and the LNRR bit to 1 simultaneously

Writing a value to this bit is disabled when the OMM0 bit of the RLN3nLMST register is $0_B$ (LIN reset mode).

Writing a value to this bit is disabled when the FTS bit is 0 (header reception or wake-up transmission/reception is stopped).
When response data of 9 bytes or more is to be transmitted or received, set this bit to 1 each time a data group (variable from 0 to 8 bytes) is transmitted or received. Once set, this bit is automatically cleared to 0 upon completion of data group transmission/reception or transition to LIN reset mode.

#### FTS Bit (LIN Communication Start)

Set this bit to 1 to start header reception or wake-up transmission/reception.
Only 1 can be written to this bit; 0 cannot be written.
Writing a value to this bit is disabled when the OMM0 bit of the RLN3nLMST register is $0_B$ (LIN reset mode).

This bit is set to 0 upon completion of frame or wake-up communication and transition to LIN reset mode.

### 19.3.3.12 RLN3nLMST — LIN Mode Status Register

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <RLIN3n_base> +$11_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | OMM1 | OMM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 19.42 RLN3nLMST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. |
| 1 | OMM1 | LIN Mode Status Monitor<br>0: LIN wake-up mode.<br>1: LIN operation mode. |
| 0 | OMM0 | LIN Reset Status Monitor<br>0: LIN reset mode.<br>1: Not in LIN reset mode. |

#### OMM1 Bit (LIN Mode Status Monitor)

The OMM1 bit indicates the current operating mode.

#### OMM0 Bit (LIN Reset Status Monitor)

The OMM0 bit indicates the current operating mode.

### 19.3.3.13 RLN3nLST — LIN Status Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $12_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | HTRC | D1RC | — | — | ERR | — | FRC | FTC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R/W | R/W |

**Table 19.43 RLN3nLST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | HTRC | Successful Header Reception Flag<br>0: Header reception has not been completed.<br>1: Header reception has been completed. |
| 6 | D1RC | Successful Data 1 Reception Flag<br>0: Data 1 reception has not been completed.<br>1: Data 1 reception has been completed. |
| 5, 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | ERR | Error Detection Flag<br>0: No error has been detected.<br>1: Error has been detected. |
| 2 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 1 | FRC | Successful Response/Wake-up Reception Flag<br>0: Response or wake-up reception has not been completed.<br>1: Response or wake-up reception has been completed. |
| 0 | FTC | Successful Response/Wake-up Transmission Flag<br>0: Response or wake-up transmission has not been completed.<br>1: Response or wake-up transmission has been completed. |

The RLN3nLST register is automatically cleared to $00_H$ upon transition to LIN reset mode.

In LIN reset mode, writing a value to this register is disabled. In LIN reset mode, the register retains $00_H$.

To clear the specific bits in the register, write 0 to the bits to be cleared and write 1 to the other bits using the store instruction.

#### HTRC Flag (Successful Header Reception Flag)

Only 0 can be written to the HTRC flag; when 1 is written, the bit retains the value before 1 is written. The HTRC flag is set to 1 upon completion of header reception. Here, an interrupt request for RLIN3n successful reception is generated if the SHIE bit in the RLN3nLIE register is 1 (interrupt is enabled). However, if header reception is completed while this bit is 1, an interrupt is not generated. To clear this bit to 0, write 0 to the bit.
To detect a new header in the response field upon completion of header reception, clear this bit after it is set to 1.

#### D1RC Flag (Successful Data 1 Reception Flag)

Only 0 can be written to the D1RC flag; when 1 is written, the bit retains the value it had before 1 is written.
The D1RC flag is set to 1 upon completion of data 1 reception. Here, an interrupt request is not

generated. Write 0 to clear this bit.
When response data of 9 bytes or more is to be received, this bit is set to 1 each time data 1 of a data group (variable from 0 to 8 bytes) is received. Write 0 before starting reception of the next data group.

### ERR Flag (Error Detection Flag)

The ERR flag is set to 1 upon detection of an error (when at least one of the flags of the RLN3nLEST register is set to 1). Here, an interrupt request for RLIN3n status is generated if the ERRIE bit in the RLN3nLIE register is 1 (interrupt is enabled). However, if an error is detected while this bit is 1, an interrupt is not generated. To clear the bit to 0, write 0 to the RPER, IPER, CSER, SFER, FER, TER, and BER flags in the RLN3nLEST register. This clears the ERR flag to 0.

### FRC Flag (Successful Response/Wake-up Reception Flag)

Only 0 can be written to the FRC flag; when 1 is written, the bit retains the value it had before 1 is written.
The FRC flag is set to 1 upon completion of response or wake-up reception. Here, an interrupt request for RLIN3n successful reception is generated if the FRCIE bit in the RLN3nLIE register is 1 (interrupt is enabled). However, if response reception or wake-up reception is completed while this bit is 1, an interrupt is not generated. Write 0 to clear this bit.
When response data of 9 bytes or more is to be received, this bit is set to 1 each time a data group (variable from 0 to 8 bytes) is received. Write 0 before starting reception of the next data group.

### FTC Flag (Successful Response/Wake-up Transmission Flag)

Only 0 can be written to the FTC flag; when 1 is written, the bit retains the value it had before 1 is written.
The FTC flag is set to 1 upon completion of response or wake-up transmission. Here, an interrupt request for RLIN3n transmission is generated if the FTCIE bit in the RLN3nLIE register is 1 (interrupt is enabled). However, if response transmission or wake-up transmission is completed while this bit is 1, an interrupt is not generated. Write 0 to clear this bit.
When response data of 9 bytes or more is to be transmitted, this bit is set to 1 each time a data group (variable from 0 to 8 bytes) is transmitted. Write 0 before starting transmission of the next data group.

### 19.3.3.14 RLN3nLEST — LIN Error Status Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $13_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RPER | IPER | CSER | SFER | FER | TER | — | BER |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W |

**Table 19.44 RLN3nLEST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RPER | Response Preparation Error Flag<br>0: Response preparation error has not been detected.<br>1: Response preparation error has been detected. |
| 6 | IPER | ID Parity Error Flag<br>0: ID parity error has not been detected.<br>1: ID parity error has been detected. |
| 5 | CSER | Checksum Error Flag<br>0: Checksum error has not been detected.<br>1: Checksum error has been detected. |
| 4 | SFER | Sync Field Error Flag<br>0: Sync field error has not been detected.<br>1: Sync field error has been detected. |
| 3 | FER | Framing Error Flag<br>0: Framing error has not been detected.<br>1: Framing error has been detected. |
| 2 | TER | Timeout Error Flag<br>0: Frame/response timeout error has not been detected.<br>1: Frame/response timeout error has been detected. |
| 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | BER | Bit Error Flag<br>0: Bit error has not been detected.<br>1: Bit error has been detected. |

The RLN3nLEST register is automatically cleared to $00_H$ upon transition to LIN reset mode.

In LIN reset mode, this register cannot be written to. In LIN reset mode, the register retains $00_H$.

To clear the specific bits in the register, write 0 to the bits to be cleared and write 1 to the other bits using the store instruction.

#### RPER Flag (Response Preparation Error Flag)

Only 0 can be written to the RPER flag; when 1 is written, the bit retains the value it had before 1 is written.

The RPER flag is set to 1 upon response preparation error detection. Write 0 to clear this bit.

#### IPER Flag (ID Parity Error Flag)

Only 0 can be written to this flag; when 1 is written, the bit retains the value it had before 1 is written.

When the IPERE bit of the RLN3nLEDE register is 1 (ID parity error detection enabled), this bit is set to 1 upon ID parity error detection. Write 0 to clear this bit.

### CSER Flag (Checksum Error Flag)

Only 0 can be written to the CSER flag; when 1 is written, the bit retains the value it had before 1 is written.
The CSER flag is set to 1 upon checksum error detection. Write 0 to clear this bit.

### SFER Flag (Sync Field Error Flag)

Only 0 can be written to this flag; when 1 is written, the bit retains the value it had before 1 is written.
When the SFERE bit of the RLN3nLEDE register is 1 (sync field error detection enabled), this bit is set to 1 upon sync field error detection. Write 0 to clear this bit.

### FER Flag (Framing Error Flag)

Only 0 can be written to the FER flag; when 1 is written, the bit retains the value it had before 1 is written.
When the FERE bit of the RLN3nLEDE register is 1 (framing error detection enabled), the FER flag is set to 1 upon framing error detection. Write 0 to clear this bit.

### TER Flag (Timeout Error Flag)

Only 0 can be written to the TER flag; when 1 is written, the bit retains the value it had before 1 is written.
When the TERE bit of the RLN3nLEDE register is 1 (frame/response timeout error detection enabled), this flag is set to 1 upon frame timeout error or response timeout error detection. Write 0 to clear this bit.

### BER Flag (Bit Error Flag)

Only 0 can be written to the BER flag; when 1 is written, the bit retains the value it had before 1 is written.
When the BERE bit of the RLN3nLEDE register is 1 (bit error detection enabled), the BER flag is set to 1 upon bit error detection. Write 0 to clear this bit.

### 19.3.3.15 RLN3nLDFC — LIN Data Field Configuration Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $14_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LSS | — | LCS | RCDS | RFDL[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.45 RLN3nLDFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | LSS | Transmission/Reception Continuation Select<br>0: The data group to be transmitted/received next is the last one.<br>1: The data group to be transmitted/received next is not the last one.<br>(Data transmission/reception continues without waiting for reception of the next header.) |
| 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5 | LCS | Checksum Select<br>0: Classic checksum mode<br>1: Enhanced checksum mode |
| 4 | RCDS | Response Field Communication Direction Select<br>0: Reception<br>1: Transmission |
| 3 to 0 | RFDL[3:0] | Response Field Length Select<br>b3 b0<br>0 0 0 0: 0 byte (+ checksum)<br>0 0 0 1: 1 byte (+ checksum)<br>0 0 1 0: 2 bytes (+ checksum)<br>:<br>0 1 1 1: 7 bytes (+ checksum)<br>1 0 0 0: 8 bytes (+ checksum)<br>Settings other than the above are prohibited. |

#### LSS Bit (Transmission/Reception Continuation Select)

The LSS bit indicates that the data group to be transmitted or received next is not the last data group when response data of 9 bytes or more is to be transmitted or received. With 0 set, data and checksum are transmitted or received because the next data group to be transmitted or received is the last one. With 1 set, only data is transmitted or received, and the checksum is not included because the next data group to be transmitted or received is not the last one.
When multi-byte response transmission/reception function is not used, set it to “0”.
Set this bit when the RTS bit is 0 (response transmission/reception stopped).

#### LCS Bit (Checksum Select)

The LCS bit selects the checksum mode.
With 0 set, classic checksum mode is selected.
With 1 set, enhanced checksum mode is selected.
When the timeout error detection is enabled (the TERE bit in the RLN3nLEDE register is 1), the specific timeout time depends on the setting of this bit. For details on the bit error, see **Section 19.7.7, Error Statuses**.
When the length of the response field data is 0 bytes (the RFDL bit is 0), do not set this bit to “1” (enhanced).

When response data of 9 bytes or more is to be transmitted or received, do not change the LCS bit setting after the first data group through the last data group.
During transmission or reception of response data of 9 bytes or more, only the last data group (the LSS bit is 0) includes the checksum, and no other groups (the LSS bit is 1) include the checksum.
Set this bit when the RTS bit is 0 (response transmission/reception stopped).

#### RCDS Bit (Response Field Communication Direction Select)

This bit selects the direction of the response field/wake-up signal communication.
With 0 set, reception is performed in the response field. In LIN wake-up mode, wake-up reception is performed (input signal low-level width count).
With 1 set, transmission is performed in the response field. In LIN wake-up mode, wake-up transmission is performed.
Set this bit when the RTS bit in the RLN3nLTRC register is 0 (response transmission/reception stopped) in LIN operation mode or when the FTS bit is 0 (header reception or wake-up transmission/reception stopped) in LIN wake-up mode).
When response data of 9 bytes or more is to be transmitted or received, do not change this bit setting after the first data group through the last data group.

#### RFDL[3:0] Bits (Response Field Length Select)

The RFDL bits set the length of the response field data.
The data length can be 0 to 8 bytes excluding the checksum size.
Set these bits when the RTS bit in the RLN3nLTRC register is 0 (response transmission/reception stopped).
When response data of 9 bytes or more is to be transmitted or received, only the last data group (the LSS bit is 0) includes the checksum, and no other groups (the LSS bit in the RLN3nLDFC register is 1) include the checksum.

### 19.3.3.16 RLN3nLIDB — LIN ID Buffer Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $15_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IDP1 | IDP0 | ID[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.46 RLN3nLIDB Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | IDP[1:0] | Parity Setting<br>Stores the parity bits (P0 and P1) received in the ID field. |
| 5 to 0 | ID[5:0] | ID Setting<br>Stores the 6-bit ID value received in the ID field. |

The value in the RLN3nLIDB register is enabled after the completion of header reception. In LIN mode (LIN operation mode, LIN wake-up mode), writing to this register is disabled.

In LIN self-test mode, the operation is as follows.

Write the value to be transmitted before communication. After completion of frame transmission/ reception (after loopback), the reversed value of the received value can be read.

For details about the LIN self-test mode, see **Section 19.9, LIN Self-Test Mode**.

#### IDP[1:0] Bits (Parity Setting)

The IDP bits store the parity bits (P0 and P1) received in the ID field of the LIN frame. IDP0 is for P0 and IDP1 is for P1.

When the IPERE bit in the RLN3nLEDE register is 1 (ID parity detection enabled), the received value and the value calculated internally are compared. If they do not match, IPER (ID parity error flag) is set.

#### ID[5:0] Bits (ID Setting)

The ID bits store the 6-bit ID value received in the ID field of the LIN frame.

### 19.3.3.17 RLN3nLCBR — LIN Checksum Buffer Register

**Access:** This register is a read-only register that can be read in 8-bit units.
However, in LIN self-test mode, this register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $16_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CKSM[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.47 RLN3nLCBR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | CKSM[7:0] | Stores the checksum value transmitted or received. |

In LIN operation mode, this register operates as follows:

- When the RCDS bit in the RLN3nLDFC register is 1 (transmission):
  The value transmitted can be read from the register. Read the value after transmission is completed.
  Writing to this register is invalid.
- When the RCDS bit in the RLN3nLDFC register is 0 (reception):
  The value received can be read from the register. Read the value after reception is completed.
  Writing to this register is invalid.

When response data of 9 bytes or more is to be transmitted or received, the checksum is appended only to the last data group; this register is not updated for the other data groups.

In LIN self-test mode, this register operates as follows:

- When the RCDS bit in the RLN3nLDFC register is 1 (transmission):
  After completion of the frame transmission (after loopback), the reversed value of the received value can be read.
- When the RCDS bit in the RLN3nLDFC register is 0 (reception):
  Write the value to be received before communication. After completion of frame transmission/reception (after loopback), the reversed value of the received value can be read.

For details about the LIN self-test mode, see **Section 19.9, LIN Self-Test Mode**.

Set the RLN3nLCBR register when the FTS bit in the RLN3nLTRC register is 0 (frame transmission or wake-up transmission/reception is stopped).

### 19.3.3.18 RLN3nLDBRb — LIN Data Buffer b Register (b = 1 to 8)

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN3nLDBR1: <RLIN3n_base> + $18_H$
RLN3nLDBR2: <RLIN3n_base> + $19_H$
RLN3nLDBR3: <RLIN3n_base> + $1A_H$
RLN3nLDBR4: <RLIN3n_base> + $1B_H$
RLN3nLDBR5: <RLIN3n_base> + $1C_H$
RLN3nLDBR6: <RLIN3n_base> + $1D_H$
RLN3nLDBR7: <RLIN3n_base> + $1E_H$
RLN3nLDBR8: <RLIN3n_base> + $1F_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LDB[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.48 RLN3nLDBRb Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | LDB[7:0] | Sets the data to be transmitted or holds the received data.<br>Setting range: $00_H$ to $FF_H$ |

- For response transmission:
  The RLN3nLDBRb registers set the data to be transmitted in the response field.
  These registers should be set when the RTS bit in the RLN3nLTRC register is 0 (response transmission/reception stop).
- For response reception:
  The RLN3nLDBRb registers store the data received in the response field.
  The received data is overwritten. If an error is detected, the data prior to reception interruption is stored in the register.
  Do not read these registers when the RTS bit is 1 (response transmission/reception is started)

In LIN self-test mode, the operation is as follows.

Write the value to be transmitted before communication. After completion of frame transmission/ reception (after loopback), the reversed value of the received value can be read.

For details about the LIN self-test mode, see **Section 19.9, LIN Self-Test Mode**.

## 19.3.4 UART Related Registers

### 19.3.4.1 RLN3nLWBR — LIN Wake-Up Baud Rate Select Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $01_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NSPB[3:0] | | | | LPRS[2:0] | | | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R |

**Table 19.49 RLN3nLWBR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | NSPB[3:0] | Bit Sampling Count Select<br>b7 b4<br>0 0 0 0: 16 samplings<br>0 1 0 1: 6 samplings<br>0 1 1 0: 7 samplings<br>0 1 1 1: 8 samplings<br>1 0 0 0: 9 samplings<br>1 0 0 1: 10 samplings<br>1 0 1 0: 11 samplings<br>1 0 1 1: 12 samplings<br>1 1 0 0: 13 samplings<br>1 1 0 1: 14 samplings<br>1 1 1 0: 15 samplings<br>1 1 1 1: 16 samplings<br>Settings other than the above are prohibited. |
| 3 to 1 | LPRS[2:0] | Prescaler Clock Select<br>b3 b1<br>0 0 0: 1/1<br>0 0 1: 1/2<br>0 1 0: 1/4<br>0 1 1: 1/8<br>1 0 0: 1/16<br>1 0 1: 1/32<br>1 1 0: 1/64<br>1 1 1: 1/128 |
| 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Configure the RLN3nLWBR register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

#### NSPB[3:0] Bits (Bit Sampling Count Select)

The NSPB bits select the number of sampling in one Tbit (reciprocal of the baud rate).
In UART mode, it is possible to set the NSPB bits from 6 samplings to 16 samplings.

#### LPRS[2:0] Bits (Prescaler Clock Select)

The LPRS bits select the frequency division ratio for the prescaler.
The LIN communication clock source is divided by this prescaler.

### 19.3.4.2 RLN3nLBRP01 — UART Baud Rate Prescaler 01 Register

**Access:** RLN3nLBRP01 register can be read or written in 16-bit units.
RLN3nLBRP0 register can be read or written in 8-bit units.
RLN3nLBRP1 register can be read or written in 8-bit units.

**Address:** RLN3nLBRP01: <RLIN3n_base> + $02_H$
RLN3nLBRP0: <RLIN3n_base> + $02_H$
RLN3nLBRP1: <RLIN3n_base> + $03_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BRP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.50 RLN3nLBRP01 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | BRP[15:0] | Assuming that the value set in this register is L (0 to 65535), the baud rate prescaler divides the frequency of the prescaler clock by L + 1.<br>Setting range: $0000_H$ to $FFFF_H$ |

Configure the RLN3nLBRP01 register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

Assuming that the value set in this register is L, the baud rate prescaler divides the frequency of the clock that is selected by the LPRS bits (prescaler clock select bits) in the RLN3nLWBR register by L + 1.

The RLN3nLBRP01 register can be accessed in 8-bit units using the registers RLN3nLBRP0 and RLN3nLBRP1.

### 19.3.4.3 RLN3nLMD — UART Mode Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $08_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | LRDNFS | — | — | — | LMD[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R | R | R | R/W | R/W |

**Table 19.51 RLN3nLMD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5 | LRDNFS | UART Reception Data Noise Filter Disable<br>0: The noise filter is enabled.<br>1: The noise filter is disabled. |
| 4 to 2 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 1, 0 | LMD[1:0] | LIN/UART Mode Select<br>b1 b0<br>0 1: UART mode |

Configure the RLN3nLMD register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

#### LRDNFS Bit (UART Reception Data Noise Filter Disable)

The LRDNFS bit enables or disables the noise filter when receiving data.
With 0 set, the noise filter is enabled when receiving data.
With 1 set, the noise filter is disabled when receiving data.

#### LMD[1:0] Bits (LIN/UART Mode Select)

The LMD bits select the LIN/UART interface mode.
To use the LIN/UART interface as an UART, set these bits to $01_B$.

### 19.3.4.4 RLN3nLBFC — UART Configuration Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $09_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | UTPS | URPS | UPS[1:0] | | USBLS | UBOS | UBLS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.52 RLN3nLBFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 6 | UTPS | UART Output Polarity Switch<br>0: Transmission data normal output<br>1: Transmission data with inverted output |
| 5 | URPS | UART Input Polarity Switch<br>0: Reception data normal output<br>1: Reception data with inverted output |
| 4, 3 | UPS[1:0] | UART Parity Select<br>00: Parity disabled<br>01: Even parity<br>10: 0 Parity<br>11: Odd parity |
| 2 | USBLS | UART Stop Bit length Select<br>0: Stop bit:1 bit<br>1: Stop bit: 2 bits |
| 1 | UBOS | UART Transfer Format Order Select<br>0: LSB First<br>1: MSB First |
| 0 | UBLS | UART Character Length Select<br>0: UART 8 bits communication<br>1: UART 7 bits communication |

Configure the RLN3nLBFC register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

#### UTPS Bit (UART Output Polarity Switch)

Sets the output polarity for UART communication.
With 0 set, transmit data is output without inversion.
With 1 set, inverted transmit data is output.
The setting of this bit is valid in all the bits of the UART frame.
In half-duplex communication, this setting should match the setting of URPS bit.

#### URPS Bit (UART Input Polarity Switch)

This bit specifies the input polarity for UART communication.
With 0 set, received data is input without inversion.
With 1 set, received data is input with inversion.
The setting of this bit is valid in all the bits of the UART frame.
In half-duplex communication, this setting should match the setting of UTPS bit.
When setting this bit to “1” and expansion bit reception ((with expansion bit comparison) or (with data

comparison)) is performed, set the inverse of the expected value to the UEBDL bit in the RLN3nLUOR1 register and RLN3nLIDB register to enable comparison of the inverted values of the received values.

### UPS[1:0] Bits (UART Parity Select)

Sets the UART parity.

- When these bits are set to "$00_B$", data is communicated without the parity.

  [Transmission]

  A parity bit is not added to transmit data.

  [Reception]

  Data is received without parity processing. Therefore, a parity error does not occur.

- When these bits are set to "$01_B$", data is communicated with the even parity.

  [Transmission]

  If the number of 1s in transmit data is odd, "1" is added to the parity bit. If the number of 1s in transmit data is even, "0" is added to the parity bit.

  [Reception]

  If the number of 1s in receive data including the parity bit is odd, a parity error occurs.

- When these bits are set to "$10_B$", data is communicated with 0 parity.

  [Transmission]

  Regardless of the number of 1s in transmit data, "0" is added to the parity bit.

  [Reception]

  The value of the parity bit is not evaluated. Therefore, no parity error occurs.

- When these bits are set to "$11_B$", data is communicated with the odd parity.

  [Transmission]

  If the number of 1s in transmit data is odd, "0" is added to the parity bit. If the number of 1s in transmit data is even, "1" is added to the parity bit.

  [Reception]

  If the number of 1s in receive data including the parity bit is even, a parity error occurs.

### USBLS Bit (UART Stop Bit Length Select)

Sets the stop bit length of data for UART communication.
With 0 set, stop bit length of 1 bit is selected.
With 1 set, stop bit length of 2 bits is selected.

### UBOS Bit (UART Transfer Format Select)

Sets the bit order of data for UART communication.
With 0 set, LSB first is selected.
With 1 set, MSB first is selected.

#### UBLS Bit (UART Character Length Select)

Sets the character length of one frame for UART communication.
With 0 set, the character length is 8 bits.
With 1 set, the character length is 7 bits.
When the character length of one frame is 9 bits (the UEBE bit in the RLN3nLUOR1 register is 1), the setting of this bit is ignored.

### 19.3.4.5 RLN3nLSC — UART Space Configuration Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $0A_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | IBS[1:0] | | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R | R | R | R |

**Table 19.53 RLN3nLSC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5, 4 | IBS[1:0] | Inter-Byte Space Select<br>b5 b4<br>0 0: 0 Tbit<br>0 1: 1 Tbit<br>1 0: 2 Tbits<br>1 1: 3 Tbits |
| 3 to 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Configure the RLN3nLSC register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

#### IBS[1:0] Bits (Inter-Byte Space Select)

The IBS bits set the width of the space between the UART frames when transmitting data from the UART buffer. 0 to 3 Tbits can be set.

Set IBS[1:0] bits to "$00_B$" when UART buffer is not used.

When data is transferred from the UART transmission data register (RLN3nLUTDR) and the UART wait transmission data register (RLN3nLUWTDR), the setting of these bits is ignored.Set these bits to "$00_B$".

### 19.3.4.6 RLN3nLEDE —UART Error Detection Enable Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $0D_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | FERE | OERE | — | BERE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R | R/W |

**Table 19.54 RLN3nLEDE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | FERE | Framing Error Detection Enable<br>0: Disables framing error detection.<br>1: Enables framing error detection. |
| 2 | OERE | Overrun Error Detection Enable<br>0: Disables overrun error detection.<br>1: Enables overrun error detection. |
| 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | BERE | Bit Error Detection Enable<br>0: Disables bit error detection.<br>1: Enables bit error detection. |

Configure the RLN3nLEDE register when the OMM0 bit in the RLN3nLMST register is $0_B$ (LIN reset mode).

#### FERE Bit (Framing Error Detection Enable)

The FERE bit enables or disables detection of the framing error.
With 0 set, the framing error is not detected.
With 1 set, the framing error is detected.
When this bit is set to 1, the detection result is reflected in the FER flag of the RLN3nLEST register.
For details on the framing error, see **Section 19.8.5, Error Statuses**.

#### OERE Bit (Overrun Error Detection Enable)

This bit enables or disables detection of the overrun error.
With 0 set, the overrun error is not detected.
With 1 set, the overrun error is detected.
When this bit is set to 1, the detection result is reflected in the OER flag of the RLN3nLEST register.
For details on the overrun error, see **Section 19.8.5, Error Statuses**.

#### BERE Bit (Bit Error Detection Enable)

The BERE bit enables or disables detection of the bit error.
With 0 set, the bit error is not detected.
With 1 set, the bit error is detected.
When this bit is set to 1, the detection result is reflected in the BER flag of the RLN3nLEST register.
In full-duplex communication, do not set this bit to “1”.
Do not set this register when the NSPB bits in the RLN3nLWBR register are $0101_B$ (6 samplings) and the LRDNFS bit in the RLN3nLMD register is 0 (noise filtering is enabled).

For details on the bit error, see **Section 19.8.5, Error Statuses**.

### 19.3.4.7 RLN3nLCUC — UART Control Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $0E_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | OM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 19.55 RLN3nLCUC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | OM0 | LIN Reset<br>0: LIN reset mode<br>1: LIN reset mode is canceled. |

After a value is written to this register, confirm that the value written is reflected in the RLN3nLMST register before writing another value.

#### OM0 Bit (LIN Reset)

The OM0 bit selects whether to transition to or exit LIN reset mode.
With 0 set, LIN reset mode.
With 1 set, LIN reset mode is canceled.

### 19.3.4.8 RLN3nLTRC — UART Transmission Control Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $10_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | RTS | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R |

**Table 19.56 RLN3nLTRC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 1 | RTS | UART Buffer Transmission Start<br>0: UART Buffer transmission is stopped.<br>1: UART Buffer transmission is started. |
| 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

#### RTS Bit (UART Buffer Transmission Start)

When transmitting data from the UART buffer, set this bit to “1”.
Only 1 can be written to this bit; 0 cannot be written.
Write to this bit when the UTOE bit in the RLN3nLUOER register is 1 (transmission enable) and the UTS bit in the RLN3nLST register is 0 (transmission is not in progress).
Once set, regardless of errors, this bit is automatically cleared to 0 upon completion of the number of data transmission specified by the MDL bit in the RLN3nLDFC register. This bit is also automatically cleared to 0 upon transition to LIN reset mode.
Writing a value to this bit is disabled when the OMM0 bit of the RLN3nLMST register is $0_B$ (LIN reset mode).
When writing 1 to this bit while the UTSW bit in the RLN3nLDFC register is 1 (when UART buffer transmission is requested, the start of transmission is delayed until the reception of stop bit is completed), write only during the reception of stop bit.

### 19.3.4.9 RLN3nLMST — UART Mode Status Register

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <RLIN3n_base> + $11_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | OMM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 19.57 RLN3nLMST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | OMM0 | LIN Reset Status Monitor<br>0: LIN reset mode.<br>1: Not in LIN reset mode. |

#### OMM0 Bit (LIN Reset Status Monitor)

The OMM0 bit indicates the current operating mode.

### 19.3.4.10 RLN3nLST — UART Status Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $12_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | URS | UTS | ERR | — | — | FTC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 19.58 RLN3nLST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5 | URS | Reception Status Flag<br>0: Reception is stopped.<br>1: Reception is started. |
| 4 | UTS | Transmission Status Flag<br>0: Transmission is stopped.<br>1: Transmission is started. |
| 3 | ERR | Error Detection Flag<br>0: No error has been detected.<br>1: Error has been detected. |
| 2, 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | FTC | Successful UART Buffer Transmission Flag<br>0: UART buffer transmission has not been completed.<br>1: UART buffer transmission has been completed. |

The RLN3nLST register is automatically cleared to "$00_H$" upon transition to LIN reset mode. In LIN reset mode, this register cannot be written to. In LIN reset mode, the register retains "$00_H$". To clear the specific bits in the register, write 0 to the bits to be cleared and write 1 to the other bits using the store instruction.

#### URS Flag (Reception Status Flag)

At the start of the reception, this flag is set to 1.
The reception is started under the following condition.

- When the start bit is detected

At the end of reception, this flag is cleared to 0. While reception is stopped, this flag retains 0.
The reception is ended under the following conditions.

- Sampling point of the first bit of the stop bits

#### UTS Flag (Transmission Status Flag)

At the start of the transmission, this flag is set to 1. During the transmission, this flag retains 1.
The transmission is started under the following conditions.

- When transmission data is specified in the RLN3nLUTDR or RLN3nLUWTDR register
- When the RTS bit in the RLN3nLTRC register is set to 1

This flag is cleared to 0 at the completion of transmission.
The transmission is ended under the following conditions.

- When transmission of data specified in the RLN3nLUTDR or RLN3nLUWTDR register is completed and next data is not specified
- When transmission from UART buffer is completed (when the RTS bit in the RLN3nLTRC register is cleared to 0)

### ERR Flag (Error Detection Flag)

This flag is set to 1 upon detection of an error, detection of an expansion bit, or when ID's match (when at least one of the flags of the RLN3nLEST register is 1). Here, an interrupt request for RLIN3n status is generated. However, if an error or expansion bit is detected or ID's match while this bit is 1, an interrupt is not generated. To clear the bit to 0, write 0 to the UPER, IDMT, EXBT, FER, OER, and BER flags in the RLN3nLEST register.

### FTC Flag (Successful UART Buffer Transmission Flag)

Only 0 can be written to the FTC flag; when 1 is written, the bit retains the value that it was before 1 was written.
Regardless of errors, this bit is set to 1 upon completion of transmission of the number of data specified by the MDL bit in the RLN3nLDFC register from the UART buffer. Here, an interrupt request for RLIN3n transmission is generated. Write 0 to clear this flag.

### 19.3.4.11 RLN3nLEST — UART Error Status Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $13_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | UPER | IDMT | EXBT | FER | OER | — | BER |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R | R/W |

**Table 19.59 RLN3nLEST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 6 | UPER | Parity Error Flag<br>0: Parity error has not been detected.<br>1: Parity error has been detected. |
| 5 | IDMT | ID Match Flag<br>0: The received data does not match the ID value.<br>1: The received data matches the ID value. |
| 4 | EXBT | Expansion Bit Detection Flag<br>0: Expansion bit has not been detected.<br>1: Expansion bit has been detected. |
| 3 | FER | Framing Error Flag<br>0: Framing error has not been detected.<br>1: Framing error has been detected. |
| 2 | OER | Overrun Error Flag<br>0: Overrun error has not been detected.<br>1: Overrun error has been detected. |
| 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | BER | Bit Error Flag<br>0: Bit error has not been detected.<br>1: Bit error has been detected. |

The RLN3nLEST register is automatically cleared to $00_H$ upon transition to LIN reset mode. In LIN reset mode, this register cannot be written to, and the value of $00_H$ is retained. To clear certain bits in this register, write 0 to those bits, and write 1 to the bits not to be cleared by using the store instruction.

#### UPER Flag (Parity Error Flag)

Only 0 can be written to this flag; when 1 is written, the bit retains the value that it was before 1 was written.
This flag is set to 1 upon parity error detection. Write 0 to clear this flag.

#### IDMT Flag (ID Match Flag)

Only 0 can be written to this flag; when 1 is written, the bit retains the value it had before 1 is written.

The IDMT flag is set to 1 when all the following conditions are met:

- The UEBE bit in the RLN3nLUOR1 register is 1 (expansion bit enabled)
- The UECD bit in the RLN3nLUOR1 register is 0 (expansion bit comparison enabled)
- The UEBDCE bit in the RLN3nLUOR1 register is 1 (expansion bit/data comparison enabled)

– The received expansion bit and the value of the UEBDL bit of the RLN3nLUOR1 register match.
– The 8-bit receive data excluding the expansion bit and the value of the RLN3nLIDB register match.

Write 0 to clear this flag.

#### EXBT Flag (Expansion Bit Detection Flag)

Only 0 can be written to this flag; when 1 is written, the bit retains the value it had before 1 is written.
When the UEBE bit in the RLN3nLUOR1 register is 1 (expansion bit enable), if the received expansion bit matches with the UEBDL bit in the RLN3nLUOR1 register, this flag is set to 1.
Write 0 to clear this flag.

#### FER Flag (Framing Error Flag)

Only 0 can be written to this flag; when 1 is written, the bit retains the value it had before 1 is written.
The FER flag is set to 1 upon framing error detection while the FERE bit of the RLN3nLEDE register is 1 (framing error detection enabled). Write 0 to clear this flag.

#### OER Flag (Overrun Error Flag)

Only 0 can be written to this flag; when 1 is written, the bit retains the value it had before 1 is written.
The OER flag is set to 1 upon overrun error detection while the OERE bit of the RLN3nLEDE register is 1 (overrun error detection enabled). Write 0 to clear this flag.

#### BER Flag (Bit Error Flag)

Only 0 can be written to the BER flag; when 1 is written, the bit retains the value it had before 1 is written.
The BER flag is set to 1 when the transmitted data and the data monitored by the receive pin do not match while the BERE bit of the RLN3nLEDE register is 1 (bit error detection enabled).
Write 0 to clear this flag.

### 19.3.4.12 RLN3nLDFC — UART Data Field Configuration Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $14_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | UTSW | — | MDL[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R | R/W | R/W | R/W | R/W |

**Table 19.60 RLN3nLDFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5 | UTSW | Transmission Start Wait<br>0: When UART buffer transmission is requested, transmission is started immediately.<br>1: When UART buffer transmission is requested, transmission is not started until reception of the stop bit is completed. |
| 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 to 0 | MDL[3:0] | UART Buffer Data Length Select<br>b3 b0<br>0 0 0 0: 9 data<br>0 0 0 1: 1 data<br>0 0 1 0: 2 data<br>0 0 1 1: 3 data<br>0 1 0 0: 4 data<br>0 1 0 1: 5 data<br>0 1 1 0: 6 data<br>0 1 1 1: 7 data<br>1 0 0 0: 8 data<br>1 0 0 1: 9 data<br>Settings other than the above are prohibited. |

#### UTSW Bit (Transmission Start Wait)

This bit controls the transmission start timing of UART buffer.
With 0 set, transmission is started as soon as the start of UART buffer transmission is requested.
With 1 set, transmission is started after the completion of the stop bit reception.
Note that the wait time is only 1 bit even if the stop bit length is set to 2 bits with the USBLS bit in the RLN3nLBFC register.
This bit is enabled when the RTS bit in the RLN3nLTRC register is set to 1. In addition, writing a value to this bit is disabled when the RTS bit is 1 (UART buffer transmission started).
Set this bit to 1 only to switch from reception to transmission in half-duplex communication.

#### MDL[3:0] Bits (UART Buffer Data Length Select)

These bits specify the data length of the UART buffer.
Writing a value to these bits is disabled when the RTS bit in the RLN3nLTRC register is 1 (UART buffer transmission started).

### 19.3.4.13 RLN3nLIDB — UART ID Buffer Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $15_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ID[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.61 RLN3nLIDB Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | ID[7:0] | Specifies the ID value that is referred in expansion bit data comparison |

#### ID Bit (ID Bit)

When the UEBE bit in the RLN3nLUOR1 register is set to 1 (expansion bit enabled), the UECD bit is set to 0 (expansion bit comparison enabled), and the UEBDCE bit is set to 1 (data comparison after expansion bit is detected), set the value to be compared with the received data. Write to the RLN3nLIDB register when the URS bit in the RLN3nLST register is 0 (reception is stopped).

### 19.3.4.14 RLN3nLUDB0 — UART Data Buffer 0 Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $17_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | UDB[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.62 RLN3nLUDB0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | UDB[7:0] | Sets the data to be transmitted.<br>Setting range: $00_H$ to $FF_H$ |

If the data length selection corresponds to 9 data bytes (RLN3nLDFC.MDL bit is "$0_H$" or "$9_H$") for multi-byte UART transmission, then the first data value for UART communication is present in this buffer.

Write to the RLN3nLUDB0 register when the RTS bit of the RLN3nLTRC register is 0 (UART buffer transmission stopped).

**Table 19.63, Bit Arrangement of the RLN3nLUDB0 Register According to Each Communication Format**, shows the bit arrangement according to the set communication format.

For details about the UART buffer, see **Section 19.8.1.2, UART Buffer Transmission**, **(1) UART Buffer Transmission**.

**Table 19.63 Bit Arrangement of the RLN3nLUDB0 Register According to Each Communication Format**

| | RLN3nLUDB0 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| 7-bit; LSB first | — | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 7-bit; MSB first | — | Bit 0 | Bit 1 | Bit 2 | Bit 3 | Bit 4 | Bit 5 | Bit 6 |
| 8-bit; LSB first | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 8-bit; MSB first | Bit 0 | Bit 1 | Bit 2 | Bit 3 | Bit 4 | Bit 5 | Bit 6 | Bit 7 |

### 19.3.4.15 RLN3nLDBRb — UART Data Buffer b Register (b = 1 to 8)

**Access:** This register can be read or written in 8-bit units.

**Address:** RLN3nLDBR1: <RLIN3n_base> + $18_H$
RLN3nLDBR2: <RLIN3n_base> + $19_H$
RLN3nLDBR3: <RLIN3n_base> + $1A_H$
RLN3nLDBR4: <RLIN3n_base> + $1B_H$
RLN3nLDBR5: <RLIN3n_base> + $1C_H$
RLN3nLDBR6: <RLIN3n_base> + $1D_H$
RLN3nLDBR7: <RLIN3n_base> + $1E_H$
RLN3nLDBR8: <RLIN3n_base> + $1F_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LDB[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.64 RLN3nLDBRb Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | LDB[7:0] | Specifies the data to be transmitted.<br>Setting range: $00_H$ to $FF_H$ |

These registers specify the data transmitted from the UART buffer.

Write to these registers when the RTS bit of the RLN3nLTRC register is 0 (UART buffer transmission stopped).

**Table 19.65, Bit Arrangement of the RLN3nLDBRb Register According to Each Communication Format**, shows the bit arrangement according to the set communication format.

For details about the UART buffer, see **Section 19.8.1.2, UART Buffer Transmission**, **(1) UART Buffer Transmission**.

**Table 19.65 Bit Arrangement of the RLN3nLDBRb Register According to Each Communication Format**

| | RLN3nLDBRb | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| 7-bit; LSB first | — | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 7-bit; MSB first | — | Bit 0 | Bit 1 | Bit 2 | Bit 3 | Bit 4 | Bit 5 | Bit 6 |
| 8-bit; LSB first | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 8-bit; MSB first | Bit 0 | Bit 1 | Bit 2 | Bit 3 | Bit 4 | Bit 5 | Bit 6 | Bit 7 |

### 19.3.4.16 RLN3nLUOER — UART Operation Enable Register

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | UROE | UTOE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 19.66 RLN3nLUOER Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 1 | UROE | Reception Enable<br>0: Disables reception.<br>1: Enables reception. |
| 0 | UTOE | Transmission Enable<br>0: Disables transmission.<br>1: Enables transmission. |

The RLN3nLUOER register is automatically cleared to $00_H$ upon transition to LIN reset mode.
In LIN reset mode, this register cannot be written to.
In LIN reset mode, the register retains $00_H$.

#### UROE Bit (Reception Enable)

The UROE bit enables or disables reception.
With 0 set, reception is disabled.
With 1 set, reception is enabled.
Do not clear this bit during reception. If the communication is suspended during reception, set the OM0 bit in the RLN3nLCUC register to 0 (LIN reset mode) to transition to the LIN reset mode. Note that transmission is also aborted.

Do not set this bit to 1 when data transmission from the UART buffer is in progress.

#### UTOE Bit (Transmission Enable)

The UTOE bit enables or disables transmission.
With 0 set, transmission is disabled.
With 1 set, transmission is enabled.
Do not clear this bit during transmission. If the communication is suspended during transmission, set the OM0 bit in the RLN3nLCUC register to 0 (LIN reset mode) to transition to the LIN reset mode. Note that reception is also aborted.

### 19.3.4.17 RLN3nLUOR1 — UART Option Register 1

**Access:** This register can be read or written in 8-bit units.

**Address:** <RLIN3n_base> + $21_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | UECD | UTIGTS | UEBDCE | UEBDL | UEBE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 19.67 RLN3nLUOR1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 5 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 4 | UECD | Expansion Bit Comparison Disable<br>0: Enables expansion bit comparison.<br>1: Disables expansion bit comparison. |
| 3 | UTIGTS | Transmission Interrupt Generation Timing Select<br>0: Transmission interrupt is generated at the start of transmission.<br>1: Transmission interrupt is generated at the completion of transmission. |
| 2 | UEBDCE | Expansion Bit Data Comparison Enable<br>0: Disables data comparison after an expansion bit is detected.<br>1: Enables data comparison after an expansion bit is detected. |
| 1 | UEBDL | Expansion Bit Detection Level Select<br>0: Selects expansion bit value 0 as the expansion bit detection level.<br>1: Selects expansion bit value 1 as the expansion bit detection level. |
| 0 | UEBE | Expansion Bit Enable<br>0: Disables expansion bit operation.<br>1: Enables expansion bit operation. |

#### UECD Bit (Expansion Bit Comparison Disable)

The UECD bit enables or disables comparison between the received expansion bit and the UEBDL bit value when the UEBE bit is 1 (expansion bit operation is enabled).
With 0 set, comparison between the received expansion bit and the UEBDL bit value is enabled when the expansion bit is received.
With 1 set, comparison between the received expansion bit and the UEBDL bit value is disabled when the expansion bit is received.
Set this bit when the OMM0 bit of the RLN3nLMST register is $0_B$ (LIN reset mode).
Do not set this bit to 1 when the UART buffer is used.
Do not set this bit to 1 when the UEBDCE bit is 1 (expansion bit/data comparison enable).

#### UTIGTS Bit (Transmission Interrupt Generation Timing Select)

The UTIGTS bit selects the generation timing of the transmission interrupt.
With 0 set, the transmission interrupt is generated at the start of transmission.
With 1 set, the transmission interrupt is generated at the completion of transmission.
When transmission from the UART buffer is performed with 0 set, the transmission interrupt is generated only at the start of the transmission of the last data (the data length is specified by the MDL bits in the RLN3nLDFC register).
When transmission from the UART buffer is performed with 1 set, the transmission interrupt is

generated only at the completion of the transmission of the last data (the data length is specified by the MDL bits in the RLN3nLDFC register).

### UEBDCE Bit (Expansion Bit Data Comparison Enable)

After an expansion bit is detected, this bit enables or disables the comparison between the 8-bit receive data excluding the expansion bit and the value of the RLN3nLIDB register.
With 0 set, when the level selected by the UEBDL bit is detected as an expansion bit, the comparison between the received value in the RLN3nLURDR register and the value of the RLN3nLIDB register is disabled.
With 1 set, when the level selected by the UEBDL bit is detected as an expansion bit, the comparison between the received value in the RLN3nLURDR register and the value of the RLN3nLIDB register is enabled.
Set this bit when the OMM0 bit of the RLN3nLMST register is $0_B$ (LIN reset mode).
Do not set this bit to 1 when the UEBE bit is 0 (expansion bit operation disabled).
Do not set this bit to 1 when the UECD bit is 1 (expansion bit comparison disabled).
Do not set this bit to 1 when the UART buffer is used.

### UEBDL Bit (Expansion Bit Detection Level Select)

The UEBDL bit selects the level to be detected as the expansion bit when the UEBE bit is 1 (expansion bit operation is enabled) and the UECD bit is 0 (expansion bit comparison enabled).
With 0 set, expansion bit value 0 is the level to be detected as the expansion bit.
With 1 set, expansion bit value 1 is the level to be detected as the expansion bit.
Set this bit when the OMM0 bit of the RLN3nLMST register is $0_B$ (LIN reset mode).
Do not set this bit to 1 when the UART buffer is used.

### UEBE Bit (Expansion Bit Enable Bit)

The UEBE bit enables or disables expansion bit operation.
With 0 set, expansion bit operation is disabled.
With 1 set, expansion bit operation is enabled.
Set this bit when the OMM0 bit of the RLN3nLMST register is $0_B$ (LIN reset mode).
Do not set this bit to 1 when the UART buffer is used.

### 19.3.4.18 RLN3nLUTDR — UART Transmission Data Register

**Access:** RLN3nLUTDR register can be read or written in 16-bit units.
RLN3nLUTDRL register can be read or written in 8-bit units.
RLN3nLUTDRH register can be read or written in 8-bit units.

**Address:** RLN3nLUTDR: <RLIN3n_base> + $24_H$
RLN3nLUTDRL: <RLIN3n_base> + $24_H$
RLN3nLUTDRH: <RLIN3n_base> + $25_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | UTD[8:0] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.68 RLN3nLUTDR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 9 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 8 to 0 | UTD[8:0] | Specifies the data to be transmitted.<br>Setting range: $000_H$ to $1FF_H$ |

The RLN3nLUTDR register specifies the data to be transmitted from the transmission data register.

Writing data to this register with the UTOE bit in the RLN3nLUOER register set to 1 starts transmission.

This register can be accessed in 8 bits.

In 9-bit communication mode, do not attempt 8-bit access.

Do not write data to this register when data transmission from the UART buffer is in progress.

Also, do not write data to this register when a transmission request is being generated due to write access to the RLN3nLUWTDR register.

When transmitting data continuously, do not set another piece of transmission data in this register before the generation of transmission interrupt.

The table below shows the bit arrangement according to the set communication format.

**Table 19.69 Bit Arrangement of the RLN3nLUTDR Register According to Each Communication Format**

| | RLN3nLUTDR | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| 7-bit; LSB first | — | — | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 7-bit; MSB first | — | — | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 |
| 8-bit; LSB first | — | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 8-bit; MSB first | — | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 | Bit7 |
| 9-bit; LSB first | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 9-bit; MSB first | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 | Bit7 | Bit8 |

### 19.3.4.19 RLN3nLURDR — UART Reception Data Register

**Access:** RLN3nLURDR register is a read-only register that can be read in 16-bit units.
RLN3nLURDRL register is a read-only register that can be read in 8-bit units.
RLN3nLURDRH register is a read-only register that can be read in 8-bit units.

**Address:** RLN3nLURDR: <RLIN3n_base> + $26_H$
RLN3nLURDRL: <RLIN3n_base> + $26_H$
RLN3nLURDRH: <RLIN3n_base> + $27_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | URD [8:0] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.70 RLN3nLURDR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 9 | Reserved | When read, the value after reset is returned. |
| 8 to 0 | URD [8:0] | Store the received data |

The RLN3nLURDR allows the reception data to be read from the receive data register.

When the UROE bit in the RLN3nLUOER register is 1, the received data is stored in this register and can be read out.

This register is updated upon reception of stop bit in the received data.

This register is also updated even when an error is caused by the parity or stop bit.

However, the value of this register is not updated if an overrun error occurs when the OERE bit of the RLN3nLEDE register is 1 (overrun detection enabled). The value of this register is updated even if an overrun error occurs when the OERE bit is 0 (overrun detection disabled).

Read this register upon occurrence of a reception error (overrun error, framing error, parity error) when the OERE bit of the RLN3nLEDE register is 1 (overrun error detection enabled). If the next data is received without reading this register, an overrun error occurs.

This register can be accessed in 8-bit units.

However, when expansion bit is used (UEBE bit of the RLN3nLUOR1 register is 1 (expansion bit operation enabled), do not attempt 8-bit access.

The table below shows the bit arrangement according to the set communication format.

**Table 19.71 Bit Arrangement of the RLN3nLURDR Register According to Each Communication Format**

| | RLN3nLURDR | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| 7-bit; LSB first | — | — | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 7-bit; MSB first | — | — | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 |
| 8-bit; LSB first | — | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 8-bit; MSB first | — | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 | Bit7 |
| 9-bit; LSB first | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 9-bit; MSB first | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 | Bit7 | Bit8 |

### 19.3.4.20 RLN3nLUWTDR — UART Wait Transmission Data Register

**Access:** RLN3nLUWTDR register can be read or written in 16-bit units.
RLN3nLUWTDRL register can be read or written in 8-bit units.
RLN3nLUWTDRH register can be read or written in 8-bit units.

**Address:** RLN3nLUWTDR: <RLIN3n_base> + $28_H$
RLN3nLUWTDRL: <RLIN3n_base> + $28_H$
RLN3nLUWTDRH: <RLIN3n_base> + $29_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | UWTD[8:0] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.72 RLN3nLUWTDR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 9 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 8 to 0 | UWTD[8:0] | Sets the data to be transmitted from the wait transmit data register after waiting for the stop bit reception to be completed.<br>Setting range: $000_H$ to $1FF_H$ |

The RLN3nLUWTDR register sets the data to be transmitted from the UART wait transmit data register.

Writing data to this register with the UTOE bit in the RLN3nLUOER register set to 1 starts transmission.

Use this register only to switch from reception to transmission in half-duplex communication.

Also, write to this register only while the stop bit is being received.

Note that the wait time is only 1 bit even if the stop bit length is set to 2 bits with the USBLS bit in the RLN3nLBFC register.

When this register is read, the value of the RLN3nLUTDR register is actually read.

In 9-bit communication mode, do not attempt 8-bit access.

Do not write data to this register when data transmission from the UART buffer is in progress.

The table below shows the bit arrangement according to the set communication format.

**Table 19.73 Bit Arrangement of the RLN3nLUWTDR Register According to Each Communication Format**

| | RLN3nLUWTDR | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| 7-bit; LSB first | — | — | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 7-bit; MSB first | — | — | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 |
| 8-bit; LSB first | — | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 8-bit; MSB first | — | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 | Bit7 |
| 9-bit; LSB first | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 9-bit; MSB first | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 | Bit7 | Bit8 |

## 19.4 Interrupt Sources

The LIN/UART interface generates four types of interrupt requests.

- RLIN3n transmission interrupt
- RLIN3n successful reception interrupt
- RLIN3n status interrupt
- RLIN3n interrupt

Setting the LIOS bit in the RLN3nLMD register to 0 allows to perform logical OR operation on all of the interrupt sources, outputting the interrupt request from the RLIN3n interrupt.

Setting the LIOS bit in the RLN3nLMD register to 1 allows to output the RLIN3n transmission interrupt, RLIN3n successful reception interrupt, or RLIN3n status interrupt depending on the interrupt request.

**Table 19.74** lists the sources for each interrupt.

**Table 19.74 Interrupt Sources**

| | | LIOS bit in RLN3nLMD register is 0 | LIOS bit in RLN3nLMD register is 1*1 | | |
|---|---|---|---|---|---|
| | | RLIN3n Interrupt | RLIN3n Transmission Interrupt | RLIN3n Successful Reception Interrupt | RLIN3n Status Interrupt |
| LIN mode | LIN master mode | • Successful frame transmission<br>• Successful frame reception<br>• Successful wake-up transmission<br>• Successful wake-up reception<br>• Successful header transmission<br>• Bit error<br>• Physical bus error<br>• Frame/response timeout error<br>• Framing error<br>• Checksum error<br>• Response preparation error | • Successful frame transmission<br>• Successful wake-up transmission<br>• Successful header transmission | • Successful wake-up reception<br>• Successful wake-up reception | • Bit error<br>• Physical bus error<br>• Frame/response timeout error<br>• Framing error<br>• Checksum error<br>• Response preparation error |
| | LIN slave mode | • Successful response transmission<br>• Successful response reception<br>• Successful wake-up transmission<br>• Successful wake-up reception<br>• Successful header reception<br>• Bit error<br>• Frame/response timeout error<br>• Framing error<br>• Sync field error<br>• Checksum error<br>• ID parity error<br>• Response preparation error | • Successful response transmission<br>• Successful wake-up transmission | • Successful response reception<br>• Successful wake-up reception<br>• Successful header reception | • Bit error<br>• Frame/response timeout error<br>• Framing error<br>• Sync field error<br>• Checksum error<br>• ID parity error<br>• Response preparation error |
| UART mode | | — | • Transmission start/ successful transmission | • Successful reception<br>• Expansion bit mismatch | • Bit error<br>• Overrun error<br>• Framing error<br>• Expansion bit match<br>• ID match<br>• Parity error |

Note 1. The LIOS bit setting is valid in LIN Mode. In UART mode, setting the LIOS bit is not required.

In LIN mode, each interrupt request is output when the corresponding bit in the RLN3nLIE register is 1 (interrupt is enabled) and the corresponding flag in the RLN3nLST register is set to 1.

## 19.5 Modes

The LIN/UART interface provides the following four modes, depending on the specific function to be performed:

- LIN reset mode
- LIN mode
  - LIN master mode
  - LIN slave mode [auto baud rate]
  - LIN slave mode [fixed baud rate]
- UART mode
- LIN self-test mode

**Figure 19.2** shows mode transitions. **Table 19.79** describes mode transition conditions. **Table 19.79** lists operations available in each mode.

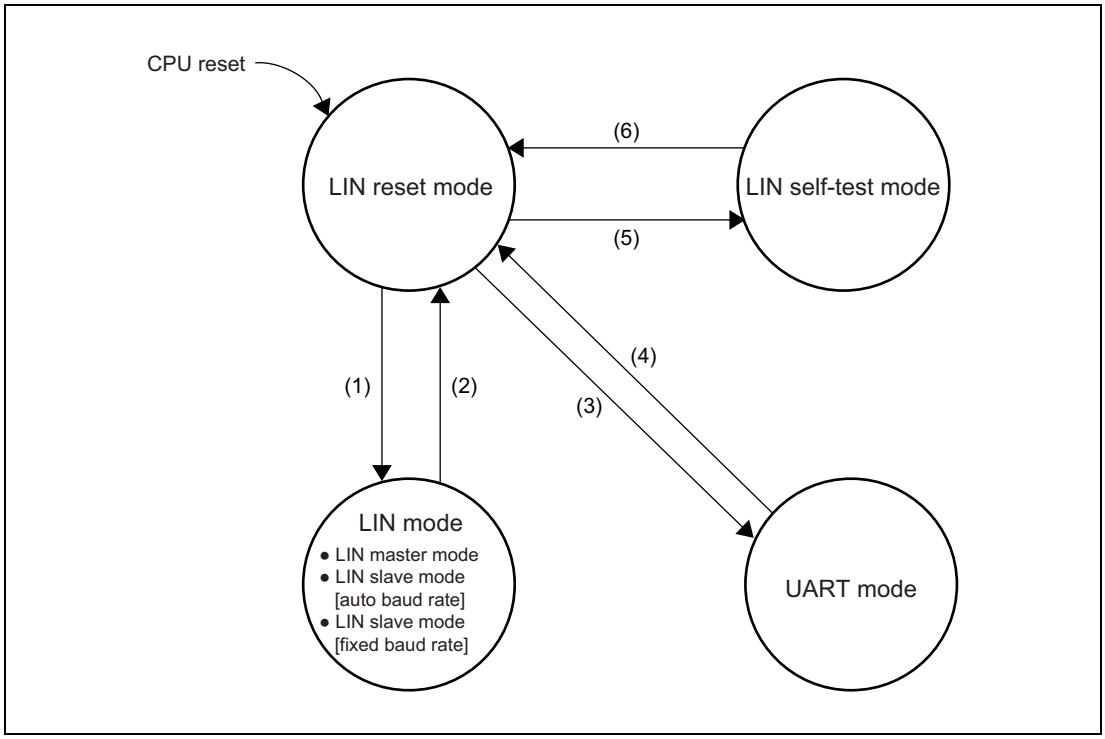



**Figure 19.2 Mode Transitions**

**Table 19.75 Transition Condition of Each Mode**

| | Mode Transition | | | Transition Condition |
|---|---|---|---|---|
| (1) | LIN reset mode | → | LIN mode | |
| | | | • LIN master mode | • LMD bits in RLN3nLMD register = $00_B$ and OM1 and OM0 bits in RLN3nLCUC register = $01_B$ or $11_B$ |
| | | | • LIN slave mode [auto baud rate] | • LMD bits in RLN3nLMD register = $11_B$ and OM1 and OM0 bits in RLN3nLCUC register = $01_B$ or $11_B$ |
| | | | • LIN slave mode [fixed baud rate] | • LMD bits in RLN3nLMD register = $10_B$ and OM1 and OM0 bits of RLN3nLCUC register = $01_B$ or $11_B$ |
| (2) | LIN mode | → | LIN reset mode | OM0 bit in RLN3nLCUC register = $0_B$ |
| (3) | LIN reset mode | → | UART mode | LMD bits in RLN3nLMD register = $01_B$ and OM0 bit in RLN3nLCUC register = $1_B$ |
| (4) | UART mode | → | LIN reset mode | OM0 bit in RLN3nLCUC register = $0_B$ |
| (5) | LIN reset mode | → | LIN self-test mode | See **Section 19.9, LIN Self-Test Mode**. |
| (6) | LIN self-test mode | → | LIN reset mode | See **Section 19.9, LIN Self-Test Mode**. |

**Table 19.76 Operations Available in Each Mode**

| LIN Mode | | | |
|---|---|---|---|
| **LIN Master Mode** | **LIN Slave Mode [auto baud rate]<br>LIN Slave Mode [fixed baud rate]** | **UART Mode** | **LIN Self-Test Mode** |
| Header transmission<br>Response transmission<br>Response reception<br>Wake-up transmission<br>Wake-up reception<br>Error detection | Header reception<br>Response transmission<br>Response reception<br>Wake-up transmission<br>Wake-up reception<br>Error detection | UART transmission<br>UART reception<br>Error detection | Self-test |

Whether mode has transitioned to LIN reset mode, the LIN mode, or the UART mode can be verified by reading the LMD bits in the RLN3nLMD register and the OMM0 bit in the RLN3nLMST register.

For a description of the LIN self-test mode, see **Section 19.9, LIN Self-Test Mode**.

## 19.6 LIN Reset Mode

Setting the OM0 bit in the RLN3nLCUC register to 0 (LIN reset mode) causes a transition to LIN reset mode. The transition to LIN reset mode can be verified by checking that the OMM0 bit in the RLN3nLMST register has been set to 0 (LIN reset mode). In this mode, the LIN communication and the UART communication functions are halted.

From LIN reset mode, transitions to LIN mode, UART mode, and LIN self-test mode can be made.

When the mode transitions to LIN reset mode, the following registers are initialized to their reset values and they retain their initial values while in LIN reset mode:

- RLN3nLTRC register
- RLN3nLST register
- RLN3nLEST register
- RLN3nLUOER register

The following registers retain their previous values even when a transition to LIN reset mode is made:

- RLN3nLWBR register
- RLN3nLBRP0 register
- RLN3nLBRP1 register
- RLN3nLMD register
- RLN3nLBFC register
- RLN3nLSC register
- RLN3nLWUP register
- RLN3nLIE register
- RLN3nLEDE register
- RLN3nLDFC register
- RLN3nLIDB register
- RLN3nLCBR register
- RLN3nLUDB0 register
- RLN3nLDBRb register (b = 1 to 8)
- RLN3nLUOR1 register
- RLN3nLUTDR register
- RLN3nLURDR register
- RLN3nLUWTDR register

## 19.7 LIN Mode

LIN mode can operate in the following submodes: LIN master mode, LIN slave mode [auto baud rate], and LIN slave mode [fixed baud rate].

In LIN master mode, the following operations can be performed: header transmission, response transmission, response reception, wake-up transmission, wake-up reception, and error detection. In LIN reset mode, setting the LMD bits in the RLN3nLMD register to $00_B$ (LIN master mode) and the OM1 and OM0 bits in the RLN3nLCUC register to either $01_B$ or $11_B$ causes the transition to LIN master mode, turning the OMM1 and OMM0 bits in the RLN3nLMST register to either $01_B$ to $11_B$.

In LIN slave mode [auto baud rate] and LIN slave mode [fixed baud rate], header reception, response transmission, response reception, wake-up transmission, wake-up reception, and error detection can be performed.

The LIN slave mode [auto baud rate] allows automatic detection of the break field and the sync field, and sets a baud rate based on the results of measurement of a sync field. The baud rate can be set to 1 kbps to 20 kbps.

Set the LPRS[2:0] bits in the RLN3nLWBR register so that the prescaler clock (the clock obtained by dividing the frequency of the LIN communication clock source by the prescaler) is configured as follows according to the target baud rate.

| [Target baud rate] | [Prescaler clock] |
|---|---|
| 1 kbps to 20 kbps | : 4MHz*1 |
| 1 kbps to 2.4 kbps (excluding 2.4 kbps) | : 4MHz |
| 2.4 kbps to 20 kbps | : 8 MHz to 12 MHz |

**Note 1.** Use the clock with NSPB[3:0] bits in the RLN3nLWBR register set to "$0011_B$" (4 samplings).

LIN slave mode [fixed baud rate] allows automatic detection of the break field, the sync field, and the ID field at a baud rate that is preset by the baud rate generator.

In LIN reset mode, setting the LMD bits in the RLN3nLMD register to $10_B$ (LIN slave mode [auto baud rate]) and setting the OM1 and OM0 bits in the RLN3nLCUC register to $01_B$ or $11_B$ causes the transition to LIN slave mode [auto baud rate]; and setting the LMD bits in the RLN3nLMD register to $11_B$ (LIN slave mode [fixed baud rate]), and setting the OM1 and OM0 bits in the RLN3nLCUC register to $01_B$ or $11_B$ causes the transition to LIN slave mode [fixed baud rate], turning the OMM1 and OMM0 bits in the RLN3nLMST register to $01_B$ or $11_B$.

When transitioning form one submode to another submode within LIN mode, transition to LIN reset mode first and change the LMD bits in the RLN3nLMD register.

The LIN mode provides the following two operating modes:

- LIN operation mode
- LIN wake-up mode

**Figure 19.3** shows the transition of operating modes. **Table 19.77** describes the transition conditions of operating modes.

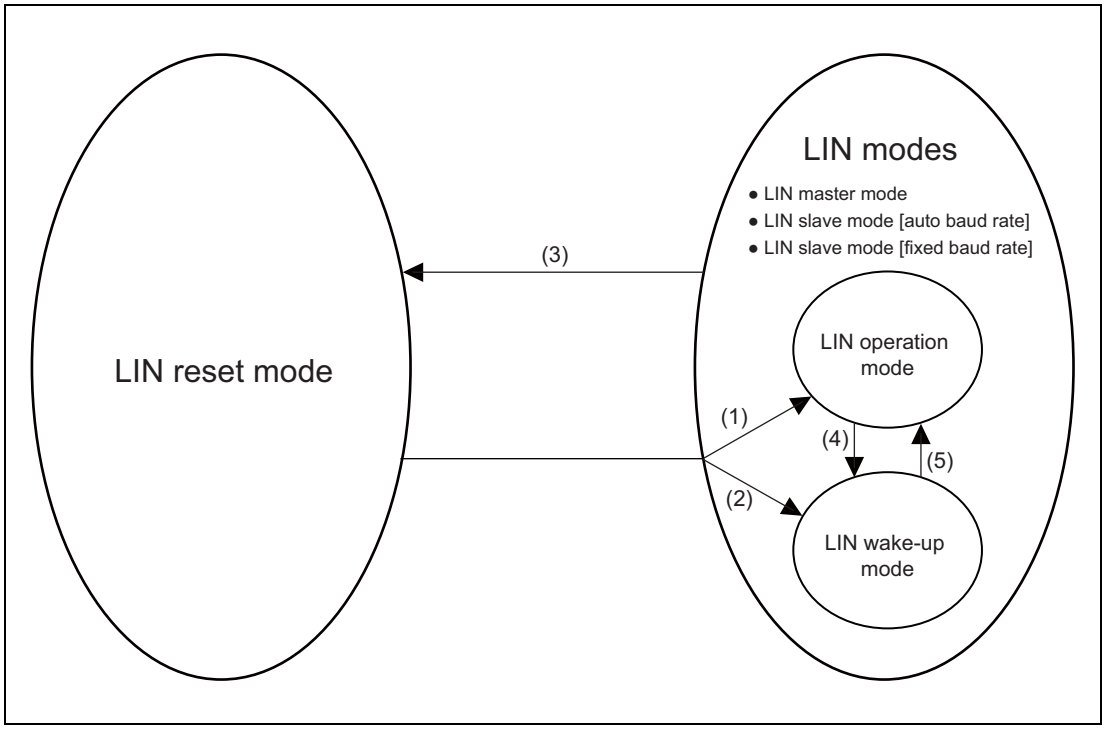



**Figure 19.3 Transition of Operating Modes**

**Table 19.77 Transition Conditions for Operating Modes**

| | Operation Mode Transition | | | Transition Condition |
|---|---|---|---|---|
| (1) | LIN reset mode | → | LIN mode<br>• LIN operation mode | LMD bits in RLN3nLMD register = $00_B$ or $10_B$ or $11_B$<br>and<br>OM1 and OM0 bits in RLN3nLCUC register = $11_B$ |
| (2) | LIN reset mode | → | LIN mode<br>• LIN wake-up mode | LMD bits in RLN3nLMD register = $00_B$ or $10_B$ or $11_B$<br>and<br>OM1 and OM0 bits in RLN3nLCUC register = $01_B$ |
| (3) | LIN mode<br>• LIN operation mode<br>• LIN wake-up mode | → | LIN reset mode | OM0 bit in RLN3nLCUC register = $0_B$ |
| (4) *1 | LIN mode<br>• LIN operation mode | → | LIN mode<br>• LIN wake-up mode | OM1 and OM0 bits in RLN3nLCUC register = $01_B$ |
| (5) *1 | LIN mode<br>• LIN wake-up mode | → | LIN mode<br>• LIN operation mode | OM1 and OM0 bits in RLN3nLCUC register = $11_B$ |

Note 1. Transition between LIN operation mode and LIN wake-up mode cannot be made when communication is in progress (when the FTS bit in the RLN3nLTRC register is 1).

### (1) LIN Operation Mode

While in LIN operation mode, frame processing (header transmission, header reception, response transmission, response reception, and error detection) can be performed.

During a transition from LIN reset mode to LIN mode, setting the OM1 and OM0 bits in the RLN3nLCUC register to $11_B$ switches the mode to LIN operation mode, changing the OMM1 and OMM0 bits in the RLN3nLMST register to $11_B$. Communication settings should be configured after the OMM1 and OMM0 bits have become $11_B$.

### (2) LIN Wake-up Mode

In LIN wake-up mode, wake-up signal processing (wake-up transmission, wake-up reception, and error detection) can be performed.

During a transition from LIN reset mode to LIN mode, setting the OM1 and OM0 bits in the RLN3nLCUC register to $01_B$ switches the mode to LIN wake-up mode, changing the OMM1 and OMM0 bits in the RLN3nLMST register to $01_B$. Communication settings should be configured after the OMM1 and OMM0 bits have become $01_B$.

## 19.7.1 LIN Master Mode

### 19.7.1.1 Header Transmission

**Figure 19.4** shows the operation of the LIN/UART interface (LIN master mode) in header transmission. **Table 19.78** provides processing in header transmission.

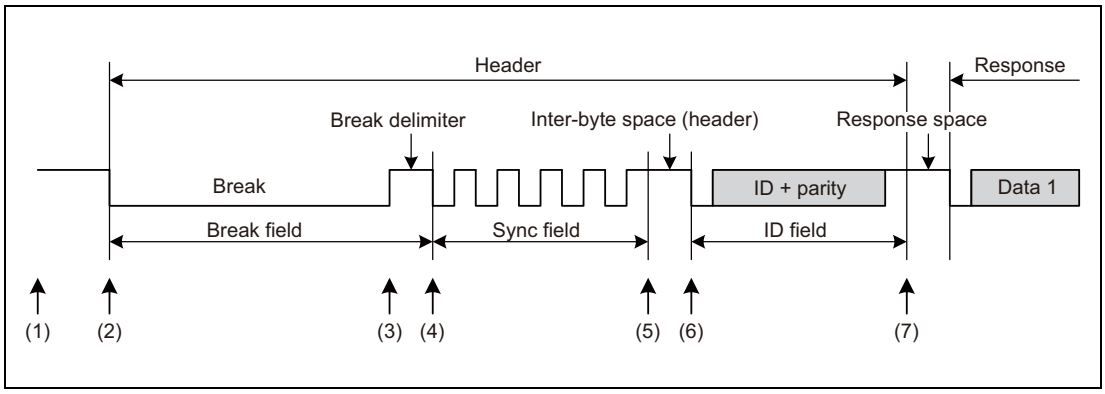



**Figure 19.4 Operation in Header Transmission**

**Table 19.78 Processing in Header Transmission**

| | Software Processing | LIN/UART Interface Processing |
|---|---|---|
| (1) | • Sets a baud rate<br>• Sets noise filter ON/OFF<br>• Enables interrupt<br>• Enables error detection<br>• Sets frame configuration parameters<br>• Transitions to LIN master mode: LIN operation mode<br>• Configures frame information to be transmitted (ID, parity, data length, response direction, Checksum method, and transmission data) | Waits for the FTS bit in the RLN3nLTRC register to be set by software (idle) |
| (2) | Sets the FTS bit in the RLN3nLTRC register to 1 (frame transmission or wake-up transmission/reception started) | Transmits a break. |
| (3) | Waits for an interrupt request | Transmits a break delimiter. |
| (4) | | Transmits a sync field ($55_H$). |
| (5) | | Transmits an inter-byte space (header). |
| (6) | | Transmits an ID field. |
| (7) | | Sets the successful header transmission flag. |

**NOTE**

For information about error detection conditions, see **Section 19.7.7, Error Statuses**.

### 19.7.1.2 Response Transmission

**Figure 19.5** shows the operation of the LIN/UART interface (LIN master mode) in response transmission. **Table 19.79** provides processing in response transmission.

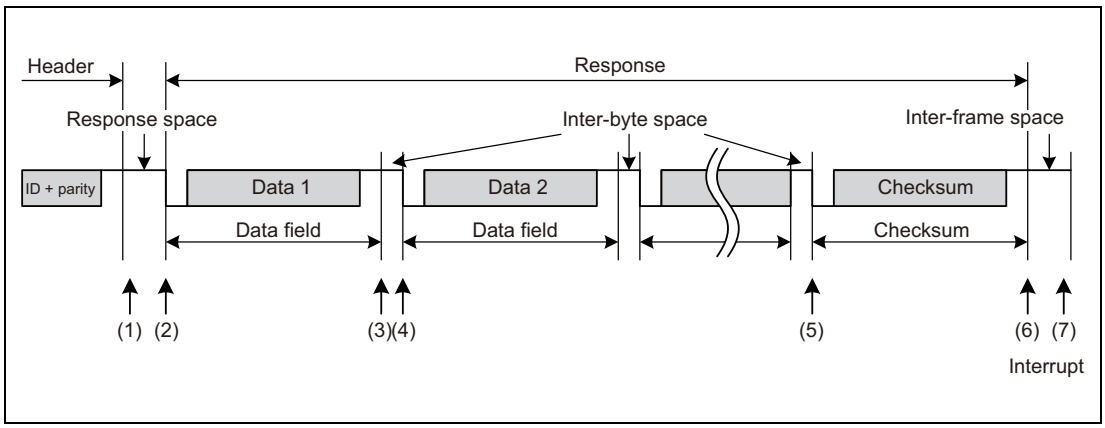



**Figure 19.5 Operation in Response Transmission**

**Table 19.79 Processing in Response Transmission**

| | Software Processing | LIN/UART Interface Processing |
|---|---|---|
| (1) | [When in frame separate mode]<br>• Sets the RTS bit in the RLN3nLTRC register to 1 (response transmission/ reception started)<br>[When not in frame separate mode]<br>• Waits for an interrupt request | [When in frame separate mode]<br>• Waits for the RTS bit in the RLN3nLTRC register to be set to 1 by software.<br>• When the bit is set to 1, transmits a response space.<br>[When not in frame separate mode]<br>• Transmits a response space. |
| (2) | Waits for an interrupt request | Transmits data 1. |
| (3) | | Transmits an inter-byte space. |
| (4) | | • Transmits data 2.<br>• Transmits an inter-byte space<br>• Transmits data 3.<br>• Transmits an inter-byte space<br>(Repeats the transmission of inter-byte spaces as many times as the data length specified in bits RFDL[3:0] in the RLN3nLDFC register).<br>:<br>: |
| (5) | | Transmits the checksum. |
| (6) | | • Sets the successful frame/wake-up transmission flag.<br>• Sets the FTS bit in the RLN3nLTRC register to 0 (frame transmission or wake-up transmission/reception stopped)<br>[When in frame separate mode]<br>• Sets the RTS bit in the RLN3nLTRC register to 0 (response transmission/reception is stopped). |
| (7) | • Processing after communication Checks the RLN3nLST register, and clears flags. | Idle |

**NOTE**

For information about error detection conditions, see **Section 19.7.7, Error Statuses**.

### 19.7.1.3 Response Reception

**Figure 19.6** shows the operation of the LIN/UART interface (LIN master mode) in response reception. **Table 19.80** provides processing in response reception.

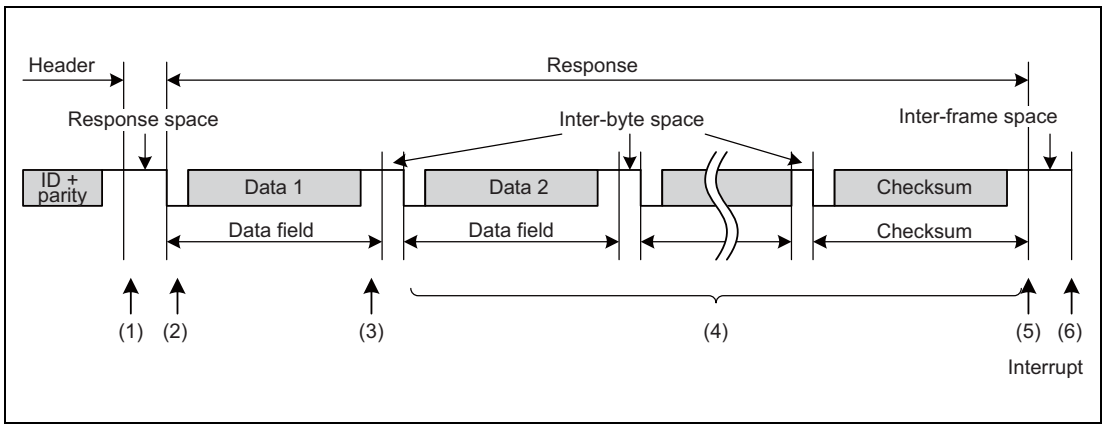



**Figure 19.6 Operation in Response Reception**

**Table 19.80 Processing in Response Reception**

| | Software Processing | LIN/UART Interface Processing |
|---|---|---|
| (1) | Waits for an interrupt request (no processing) | Waits for detection of a start bit. |
| (2) | | Receives data 1 when the start bit is detected. |
| (3) | | Sets the successful data 1 reception flag. |
| (4) | | • Receives data 2 when the start bit is detected.<br>• Receives data 3 when the start bit is detected.<br>Repeats the transmission of inter-byte spaces as many times as the data length specified in bits RFDL[3:0] in the RLN3nLDFC register).<br>:<br>:<br>• Receives the checksum when the start bit is detected. |
| (5) | | • Determines the checksum.<br>• Sets the successful frame/wake-up reception flag.<br>• Sets the FTS bit in the RLN3nLTRC register to 0 (frame transmission or wake-up transmission/reception stopped). |
| (6) | • Processing after communication<br>Reads the received data.<br>Checks the RLN3nLST register, and clears flags. | Idle |

**NOTE**

For information about error detection, see **Section 19.7.7, Error Statuses**.

## 19.7.2 LIN Slave Mode

### 19.7.2.1 Header Reception

**Figure 19.7** shows the operation of the LIN/UART interface (LIN slave mode) in header reception.
**Table 19.81** provides processing in header reception.

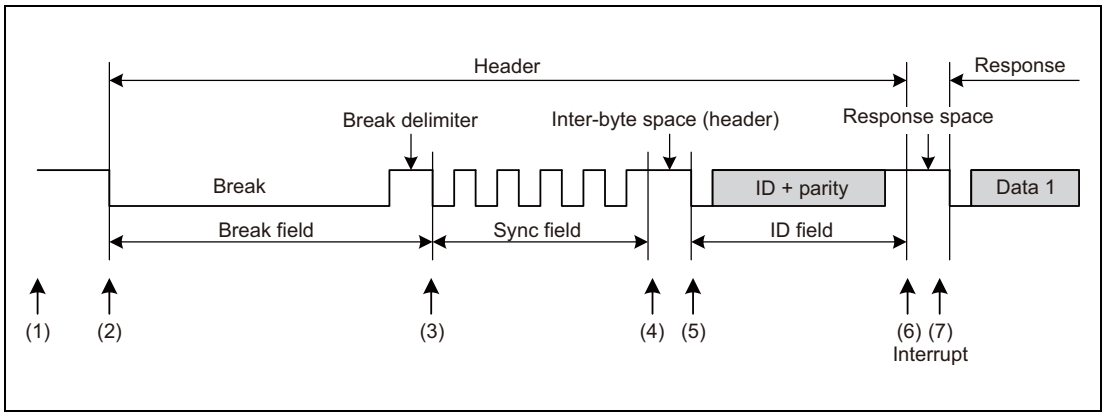



**Figure 19.7 Operation in Header Reception**

**Table 19.81 Processing in Header Reception**

| | Software Processing | LIN/UART Interface Processing |
|---|---|---|
| (1) | • Sets a baud rate<br>• Sets noise filter ON/OFF<br>• Enables interrupt<br>• Enables error detection<br>• Sets frame configuration parameters<br>• Transitions to the LIN slave mode: LIN operation mode<br>• Sets the FTS bit in the RLN3nLTRC register to 1 (header reception or wake-up transmission/ reception started) | Waits for the FTS bit in the RLN3nLTRC register to be set by software. |
| (2) | Waits for an interrupt request. | Waits for detection of break field |
| (3) | | Detects a break field (LIN slave mode [fixed baud rate]). For details about the break field detection timing in the case of LIN slave mode [auto baud rate], see [Auto Baud Rate Correction Function].) |
| (4) | | • Detects a sync field ($55_H$)<br>• Sets the baud rate generator (in the case of LIN slave mode [auto baud rate])<br>• Clears the no-response request bit (LNRR bit). |
| (5) | | • Receives an ID field.<br>• Checks the ID parity bit |
| (6) | | Sets the header reception complete flag. |
| (7) | • Checks the RLN3nLST register, and clears flags.<br>• Checks the RLN3nLIDB register, and prepares a response. | • Completes a header reception process.<br>• Waits for a response request. |

**NOTE**

The LIN/UART interface allows reception of break fields during frame transmission/reception. In that case, a framing error, bit error or other error may be detected at the stop bit position of the frame before the break field is received, and a status interrupt may occur as a result. However, reception of a new header (the subsequent Sync field and ID field) continues regardless of whether an error occurred. For information about error detection conditions, see **Section 19.7.7, Error Statuses**.

### [Auto Baud Rate Correction Function]

In LIN slave mode [auto baud rate], the system always measures the low-level widths that are received. If the first "Low level" width is 10 times (if the BLT bit of the RLN3nLBFC register is "0") or 11 times (if the BLT bit of the RLN3nLBFC register is "1") or greater than the bit width calculated from the average of the starting 2 bits (the period of the consecutive falling edges from the beginning of the sync field) of the sync field, the system concludes that the detection of break field was successful and verifies that the data in the sync field is $55_H$. If the data in the sync field is indeed $55_H$ and the system determines that sync field reception was successful, the system automatically sets the baud rate correction result to the RLN3nLBRP01 register.

If data is received up to the ID field without error, a successful header reception interrupt is generated at the stop bit position.

On the other hand, if the data in the sync field is not $55_H$ and the system determines that sync field reception failed, the system sets the sync field error flag and an error interrupt is generated. In that case, baud rate correction is not performed and the LIN/UART interface waits for the detection of the next break field (low level).

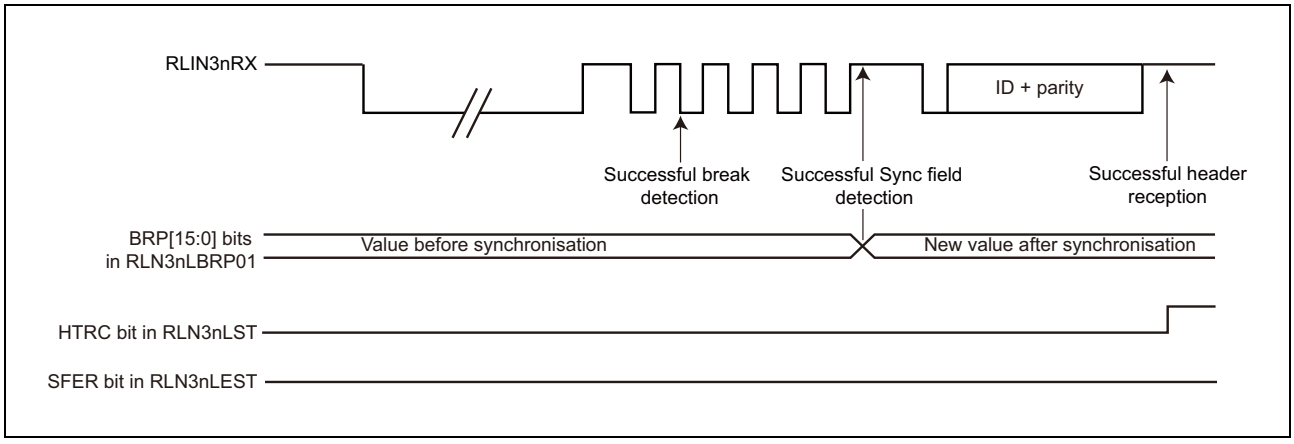



**Figure 19.8 Header Reception in LIN Slave Mode [Auto Baud Rate] (in Normal Operation)**

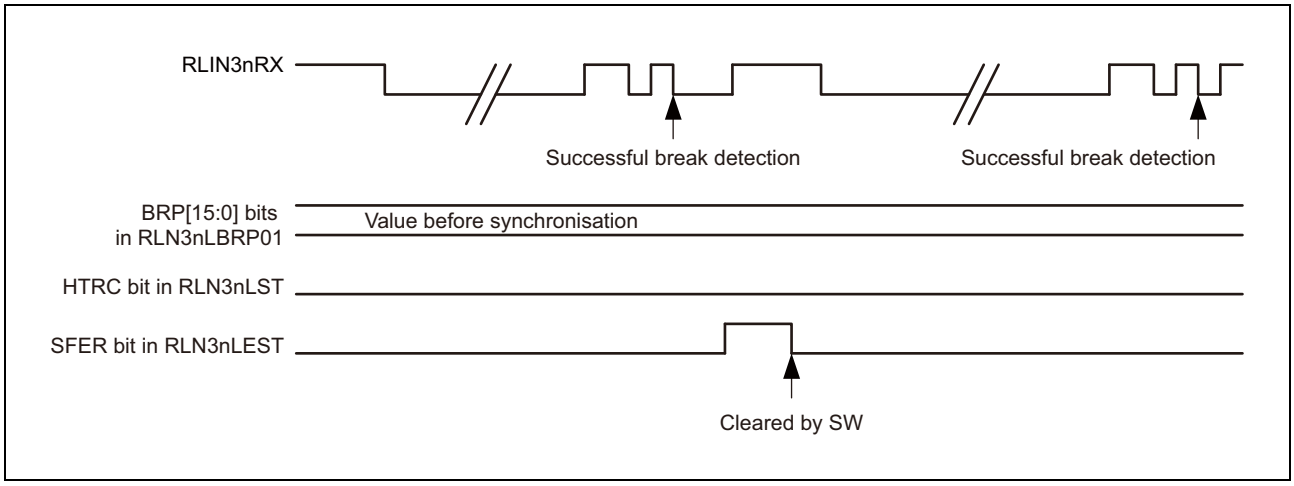



**Figure 19.9 Header Reception in LIN Slave Mode [Auto Baud Rate] (Sync Field Error)**

### 19.7.2.2 Response Transmission

**Figure 19.10** shows the operation of the LIN/UART interface (LIN slave mode) in response transmission. **Table 19.82** provides processing in response transmission.

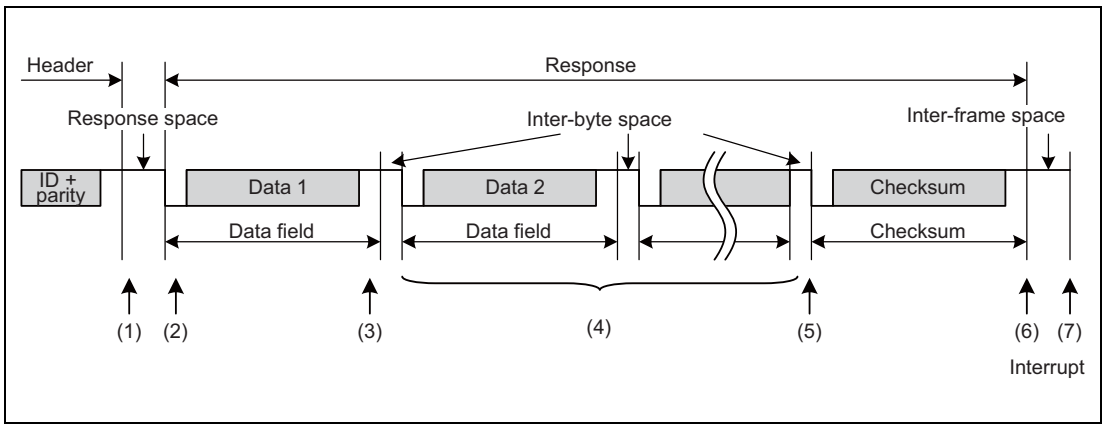



**Figure 19.10 Operation in Response Transmission**

**Table 19.82 Processing in Response Transmission**

| | Software Processing | LIN/UART Interface Processing |
|---|---|---|
| (1) | • Configures the RLN3nLDFC register.<br>• Configures the RLN3nLDBRb registers.(b = 1 to 8)<br>• Sets the RTS bit in the RLN3nLTRC register to 1 (response transmission/ reception started) | • Waits for the RTS or LNRR bit in the RLN3nLTRC register to be set by software<br>• Transmits the response space after the RTS bit of the RLN3nLTRC register is set to 1 |
| (2) | Waits for an interrupt request. | Transmits data 1. |
| (3) | | Transmits an inter-byte space. |
| (4) | | • Transmits data 2.<br>• Transmits an inter-byte space<br>• Transmits data 3.<br>• Transmits an inter-byte space<br>(Repeats as many times as the data length specified in bits RFDL[3:0] in the RLN3nLDFC register).<br>:<br>: |
| (5) | | Transmits the checksum. |
| (6) | | • Sets the successful response/wake-up transmission flag.<br>• Sets the RTS bit in the RLN3nLTRC register to 0 (response transmission/reception stopped). |
| (7) | • Processing after communication Checks the RLN3nLST register, and clears flags. | • Completes the response transmission process.<br>• Waits for a new break. |

**NOTE**

- For information about error detection, see **Section 19.7.7, Error Statuses**.
- The LIN/UART interface allows reception of break fields during frame transmission/reception. In that case, a framing error, bit error or other error may be detected at the stop bit position of the frame before the break field is received, and a status interrupt may occur as a result. However, reception of a new header (the subsequent Sync field and ID field) continues regardless of whether an error occurred.

### 19.7.2.3 Response Reception

**Figure 19.11** shows the operation of the LIN/UART interface (LIN slave mode) in response reception. **Table 19.83** provides processing in response reception.

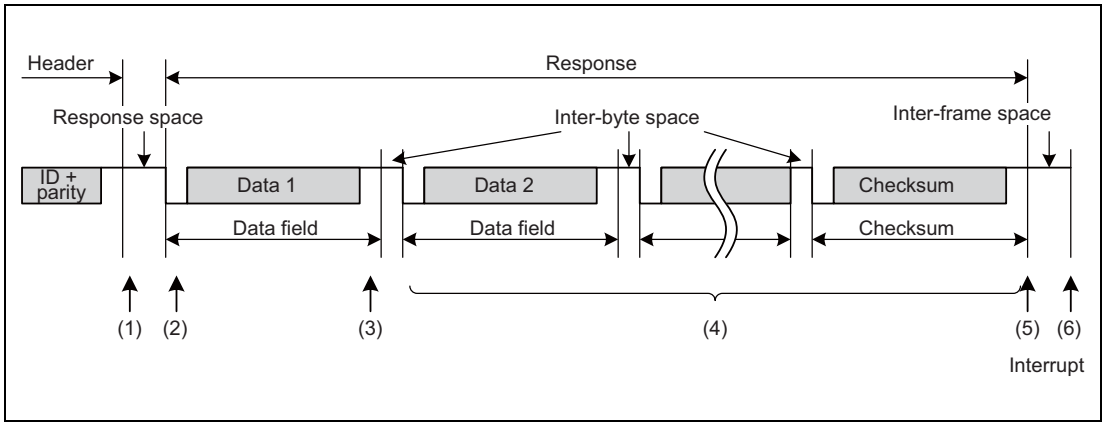



**Figure 19.11 Operation in Response Reception**

**Table 19.83 Processing in Response Reception**

| | Software Processing | LIN/UART Interface Processing |
|---|---|---|
| (1) | • Sets the RLN3nLDFC register.<br>• Sets the response transmission/reception start bit (RTS bit) to 1. | • Waits for the RTS (response transmission/reception start) or LNRR (no-response request) bit to be set by software.<br>• Waits for detection of the start bit. |
| (2) | Waits for an interrupt request. | Receives data 1 when the start bit is detected. |
| (3) | | Sets the successful data 1 reception flag. |
| (4) | | • Receives data 2 when the start bit is detected.<br>• Receives data 3 when the start bit is detected.<br>Repeats as many times as the data length specified in bits RFDL[3:0] in the RLN3nLDFC register).<br>⋮<br>⋮<br>• Receives the checksum when the start bit is detected. |
| (5) | | • Determines the checksum.<br>• Sets the successful response/wake-up reception flag or error flag.<br>• Sets the RTS bit in the RLN3nLTRC register to 0 (response transmission/reception stopped). |
| (6) | • Processing after communication<br>Reads the received data.<br>Checks the RLN3nLST register, and clears flags. | • Completes the response process.<br>• Waits for a new break. |

**NOTE**

- For information about error detection conditions, see **Section 19.7.7, Error Statuses**.
- The LIN/UART interface allows reception of break fields during frame transmission/reception. In that case, a framing error, bit error or other error may be detected at the stop bit position of the frame before the break field is received, and a status interrupt may occur as a result. However, reception of a new header (the subsequent Sync field and ID field) continues regardless of whether an error occurred.

### 19.7.2.4 No-response Request

**Figure 19.12** shows the operation of the LIN/UART interface (LIN slave mode) when no response is requested. **Table 19.84** shows the processing that occurs when no response is requested.

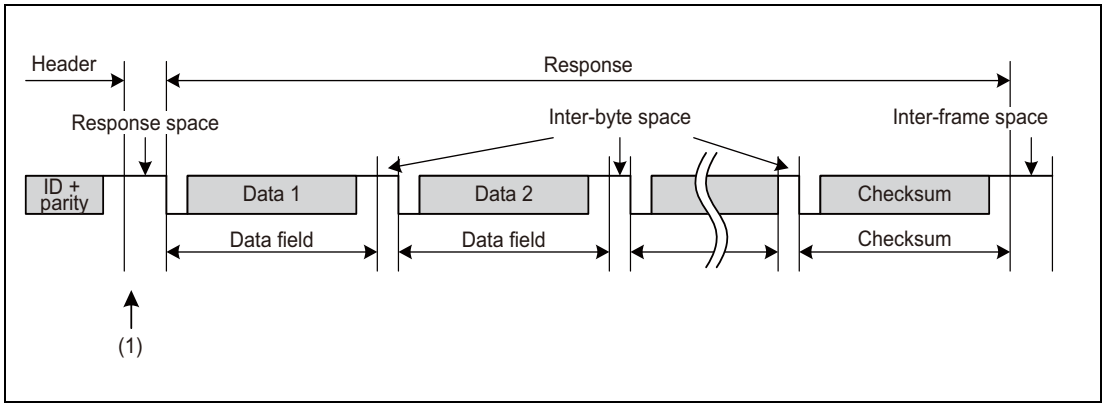



**Figure 19.12 Operation when No Response is Requested**

**Table 19.84 Processing when No Response is Requested**

| Software Processing | LIN/UART Interface Processing |
|---|---|
| (1) | • Waits for the LNRR (no-response request) bit to be set by software |
| • Sets the no-response request bit (LNRR bit) to 1. | • Completes the frame reception process |
| | • Waits for a new break |

## 19.7.3 Data Transmission/Reception

### 19.7.3.1 Data Transmission

One bit of data is transmitted per 1 Tbit.

The data that is transmitted returns to the reception data input pin via the LIN transceiver. The received data and the transmitted data are compared bit by bit, and the results are stored in the BER flag of the RLN3nLEST register (see **Section 19.7.7, Error Statuses**).

In LIN master mode and LIN slave mode [fixed baud rate], 1 Tbit is generated to be $16f_{LIN}$, and thus the sampling point for received data is at the 13th clock cycle (81.25% position).

In LIN slave mode [auto baud rate], if 1 Tbit is generated to be $4f_{LIN}$, the sampling point for received data is at the third clock cycle (75% position). If 1 Tbit is generated to be $8f_{LIN}$, the sampling point for received data is at the 7th clock cycle (87.5% position).

**Figure 19.13** shows an example of data transmission timing.

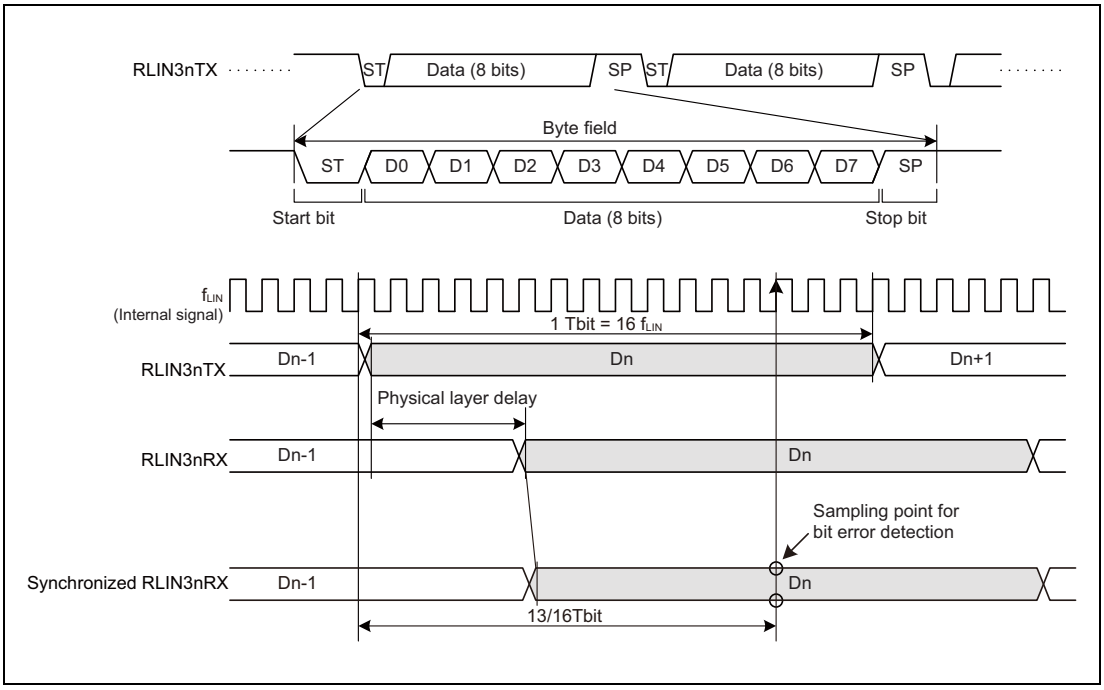



**Figure 19.13 Example of Data Transmission Timing (LIN Master Mode, LIN Slave Mode [Fixed Baud Rate])**

### 19.7.3.2 Data Reception

Data reception is performed by using the synchronized RLIN3nRX signal (an internal signal) that is the input from the RLIN3nRX pin synchronized with prescaler clock.

The byte field is synchronized at the falling edge of the start bit for the synchronized RLIN3nRX signal. After the falling edge is detected, sampling is performed again 0.5 Tbit later, and the falling edge is recognized as a start bit if the synchronized RLIN3nRX signal is low level. The falling edge is not recognized as a start bit if the RLIN3nRX signal after the reset is de-asserted is fixed to low level or if a high level is detected on re-sampling.

After the start bit is detected, the system samples 1 bit per Tbit.

The LIN/UART interface has a noise filter function with respect to reception data. If the LRDNFS bit in the RLN3nLMD register is 0, the LIN/UART interface uses a noise filter, and the value determined by a 3-sampling majority rule on prescaler clocks is used as the sampling value. If the LRDNFS bit in the RLN3nLMD register is 1, the LIN/UART interface does not use a noise filter, and the value of the synchronized RLIN3nRX value at the sampling position is used as the sampling value.

**Figure 19.14** shows an example of data reception timing.

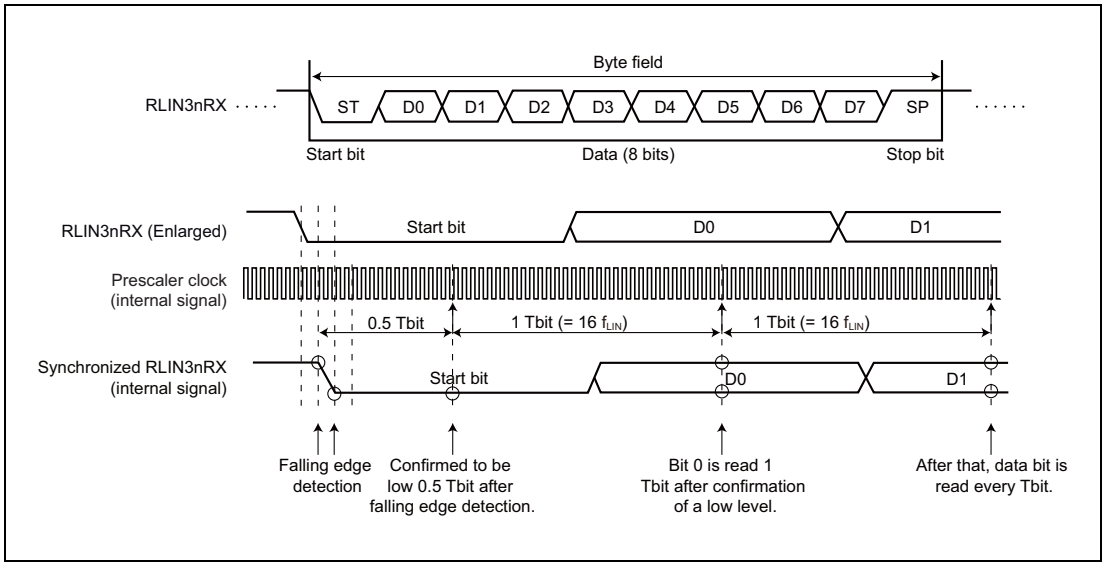



**Figure 19.14 Example of Data Reception Timing (LIN Master Mode, LIN Slave Mode [Fixed Baud Rate])**

## 19.7.4 Transmission/Reception Data Buffering

This section explains the buffer processing that takes place when the LIN/UART interface transmits or receives data continuously.

### 19.7.4.1 Transmission of LIN Frames

For an 8-byte transmission, the contents stored in registers RLN3nLDBR1 to RLN3nLDBR8 are sequentially transmitted to data regions 1 to 8 of the LIN frame. In the case of a 4-bytes transmission, the contents stored in registers RLN3nLDBR1 to RLN3nLDBR4 are transmitted to data regions 1 to 4 of the LIN frame, but the contents of registers RLN3nLDBR5 to RLN3nLDBR8 are not transmitted. The transmitted checksum data is stored in the RLN3nLCBR register.

**Figure 19.15** shows the LIN transmission processing and the corresponding buffers.

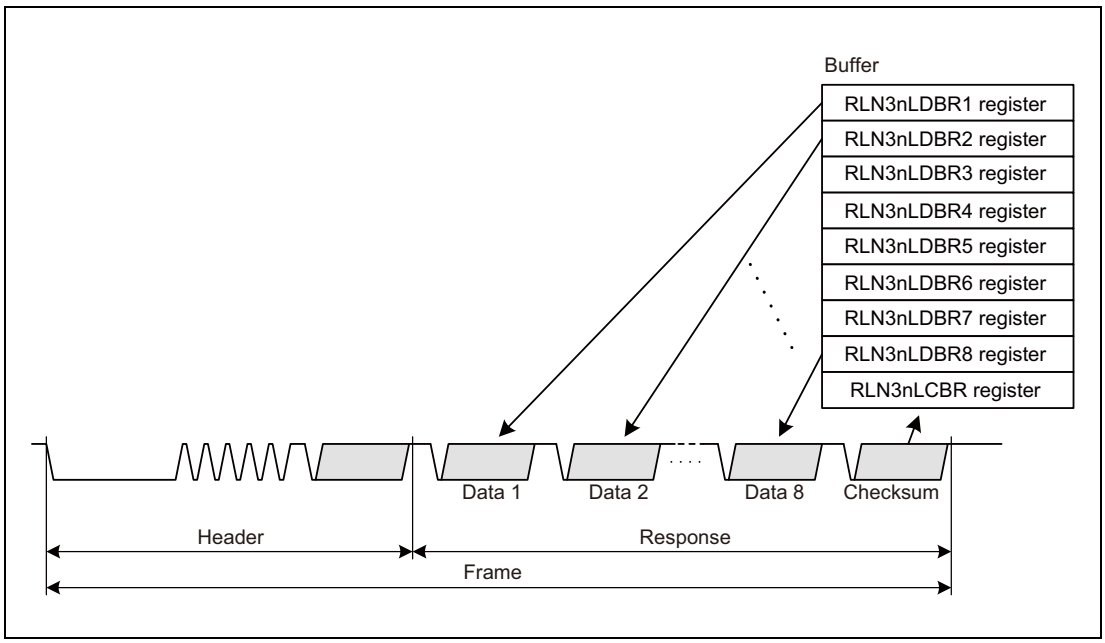



**Figure 19.15 LIN Transmission Processing and Corresponding Buffers**

**[Frame Separate Mode]**

Setting the FSM bit in the RLN3nLDFC register to 1 turns on the frame separate mode.

In frame separate mode, the header and response are separately transmitted when prompted by respective transmission start requests.

When the transmission of the header is completed, the HTRC flag in the RLN3nLST register is set to 1 (successful header transmission).

Use frame separate mode when transmitting or receiving response data of 9 bytes or greater in LIN master mode.

### 19.7.4.2 Reception of LIN Frames

For an 8-byte reception, the contents of data regions 1 to 8 of the LIN frame are stored in registers RLN3nLDBR1 to RLN3nLDBR8, respectively, upon reception of a stop bit. In the case of a 4-byte reception, the contents of data regions 1 to 4 of the LIN frame are stored in registers RLN3nLDBR1 to RLN3nLDBR4, respectively; no data is stored in registers RLN3nLDBR5 to RLN3nLDBR8. Also, the received checksum data is stored in the RLN3nLCBR register.

**Figure 19.16** depicts the LIN reception processing and the corresponding buffers.

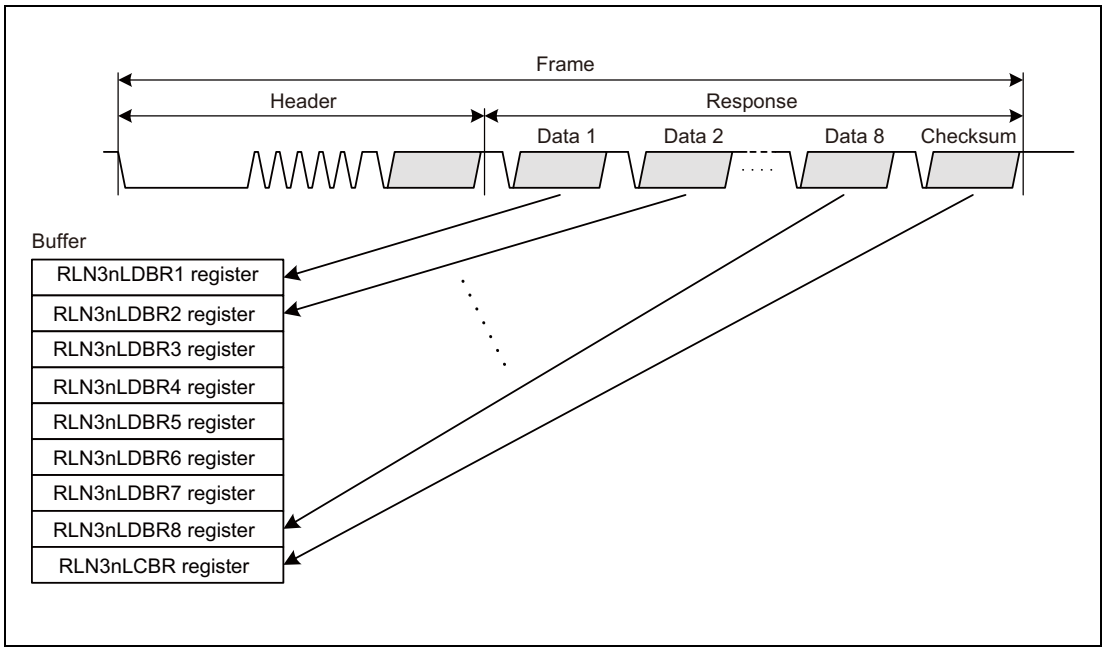



**Figure 19.16 LIN Reception Processing and Corresponding Buffers**

**[Reception of Data 1]**

When the reception of the first byte of data is completed, the D1RC flag in the RLN3nLST register is set to 1 (successful data 1 reception).

#### 19.7.4.3 Multi-Byte Response Transmission/Reception Function

Normally in LIN communications, a response is 9 bytes or less including a checksum field; however, responses of 10 bytes or greater can also be transmitted and received.

In such case, the bit error, framing error, response preparation error detection, and auto checksum functions are enabled.

If the data length is greater than 8 bytes, the LSS bit in RLN3nLDFC register should be set to 1 (indicating that the next data group to be transmitted or received is not the final data group) in the first data group (variable from 0 to 8 bytes) before transmitting or receiving the data group. After the transmission or reception, the user should determine whether the next data group is the final data group. If it is the final data group, the LSS bit in the RLN3nLDFC register should be set to 0 (indicating that the next data group to be transmitted or received is the final data group) before transmitting or receiving the data group, and a checksum should be appended to the final data group.

By changing the RFDL bit setting in RLN3nLDFC register when the RTS bit in RLN3nLTRC register is 0, the user can change the data length for each data group.

When performing multi-byte response transmission/reception in LIN master mode, set the FSM bit in RLN3nLDFC register to 1 (frame separate mode).

**NOTE**

In LIN slave mode, the LIN/UART interface can detect a new break field during the transmission or reception of a response.

### 19.7.5 Wake-up Transmission/Reception

The wake-up transmission/reception can be used in LIN wake-up mode.

#### 19.7.5.1 Wake-up Transmission

In LIN wake-up mode, setting the RFT bit in the RLN3nLDFC register to 1 (LIN master mode: response transmission), or setting the RCDS bit in the RLN3nLDFC register to 1 (LIN slave mode: response transmission), and then the FTS bit in the RLN3nLTRC register to 1 (frame transmission, header reception or wake-up transmission/reception started) causes a wake-up signal to be output from the output pin. The low level width of the wake-up signal is set using the WUTL[3:0] bits in the RLN3nLWUP register.

However, if the LWBR0 bit of the RLN3nLWBR register in LIN master mode is 1 (LIN2.x), the LIN system clock ($f_{LIN}$) becomes low level width at fa regardless of the setting of the LCKS bit of the RLN3nLMD register. By setting the WUTL[3:0] bits of the RLN3nLWUP register to $0100_B$ (5 Tbits), 260 µs low width can be output in LIN wake-up mode regardless of the setting of the LCKS bit of the RLN3nLMD register.

If a wake-up low-level width is output without any bit error, the FTC flag in the RLN3nLST register is set to 1 (successful frame response or wake-up transmission) and when the FTCIE bit in the RLN3nLIE register is 1 (successful frame response/wake-up transmission interrupt enabled), an interrupt request for RLIN3n transmission is generated.

If RLN3nLEDE.BERE is set and a bit error is detected, wake-up transmission is aborted and the BER flag of the RLN3nLEST register is set to 1 (bit error detected).
When RLN3nLEDE.PBERE is set in LIN master mode, set RLN3nLEST.PBER flag to 1 (physical bus error detection) at the same time of a bit error.

**Figure 19.17** shows the wake-up transmission timing.

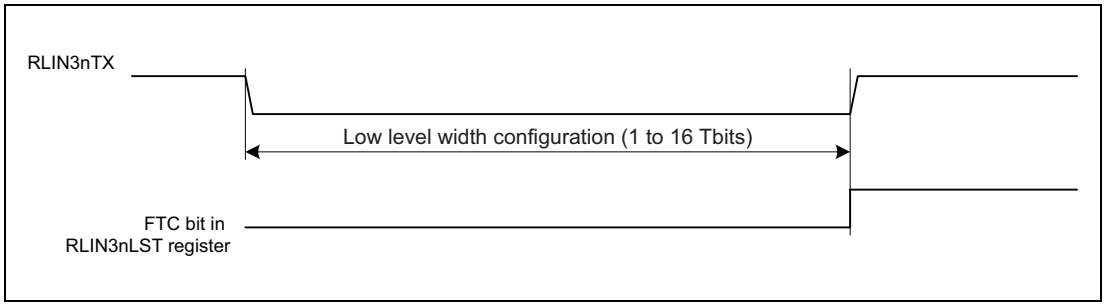



**Figure 19.17 Wake-up Transmission Timing**

### 19.7.5.2 Wake-up Reception

To detect a wake-up signal, use the input signal low-level width count function. The input signal low level width count function measures the low level width of the input signal to the RLIN3nRX pin, using the same sampling point as data reception. This allows the 2.5-Tbit or longer low-level width of the input signal of $f_{LIN}$ to be measured.

In LIN master mode, by setting the LWBR0 bit in the RLN3nLWBR register, operation is executed without changing the baud rate generator setting at a transition between LIN operation mode and LIN wake-up mode.

When LIN Specification Package Revision 1.3 is used, set the LWBR0 bit in the RLN3nLWBR register to 0. When LIN Specification Package Revision 2.x is used, set the LWBR0 bit to 1. Setting the LWBR0 bit to 1 selects the LIN system clock ($f_{LIN}$) to fa regardless of the setting of the LCKS bit in the RLN3nLMD register. (The LCKS bit is not changed). By setting the baud rate to 19200bps while fa is selected, an input signal with a low-level width of 130 us or longer regardless of the setting of the LCKS bit in the RLN3nLMD register.

To used the wake-up reception function, in LIN wake-up mode, set the RFT bit in the RLN3nLDFC register to 0 (LIN master mode: response reception) or the RCDS bit in the RLN3nLDFC register to 0 (LIN slave mode: response reception), and then set the FTS bit in the RLN3nLTRC register to 1 (frame transmission (header reception) or wake-up transmission/reception started).

When the low level width to be measured is reached, the FRC flag in the RLN3nLST register is set to 1 (successful frame response/wake-up reception) and if the FRCIE bit in the RLN3nLIE register is 1 (successful frame response or wake-up reception interrupt enabled), an interrupt request for RLIN3n successful reception is generated.

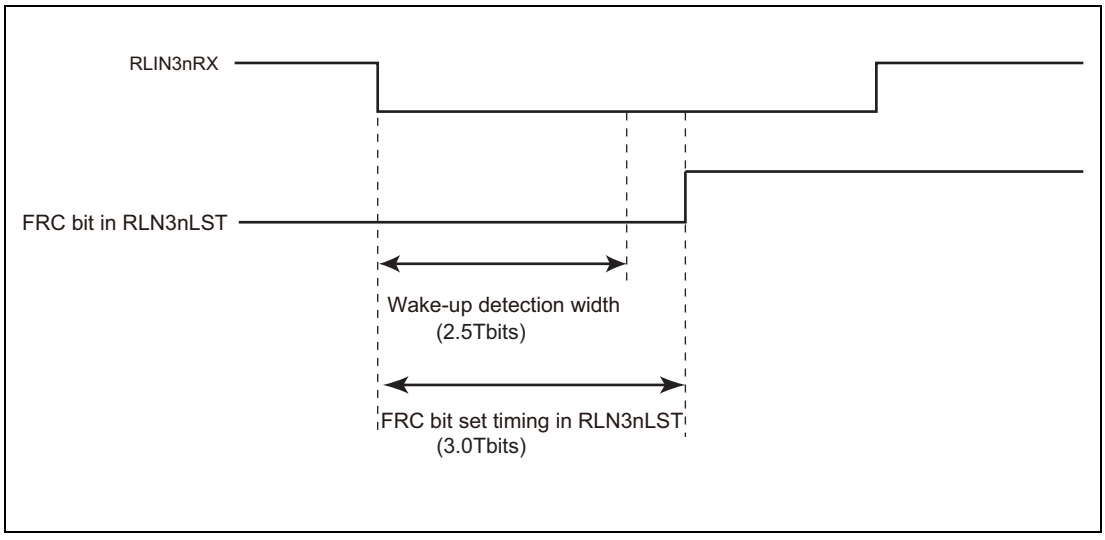



**Figure 19.18 Input Signal Low level Count Function**

### 19.7.5.3 Wakeup Collision

If the master node and the slave node transmit wakeup signals simultaneously, a collision will occur on the LIN bus, however the collision of wakeup signals is not detected in the LIN/UART interface.

### 19.7.6 Statuses

During LIN mode operation, the LIN/UART interface can detect seven types of statuses.

The four statuses, successful frame/wake-up transmission, successful frame/wake-up reception, error detection, successful header transmission/reception, can generate interrupt requests.

**Table 19.85** shows the types of statuses available in LIN master mode. **Table 19.86** lists the types of statuses available in LIN slave mode [auto baud rate] and in LIN slave mode [fixed baud rate].

**Table 19.85 Types of Statuses in LIN Master Mode**

| Status | Status Set Condition | Status Clear Condition | Operating Mode Capable of Status Detection | Corresponding Bit | Interrupt |
|---|---|---|---|---|---|
| Reset | After the OM0 bit in the RLN3nLCUC register is set to not-LIN-reset-mode, if the LIN/UART interface actually exits LIN reset mode. | After the OM0 bit in the RLN3nLCUC register is set to LIN reset mode, if the LIN/UART interface actually enters LIN reset mode. | All modes | OMM0 bit in RLN3nLMST register | — |
| Operation mode | After the OM1 bit in the RLN3nLCUC register is set to LIN operation mode, if the LIN/UART interface actually enters LIN operation mode. | After the OM1 bit in the RLN3nLCUC register is set to LIN wake-up mode, if the LIN/UART interface actually enters LIN wake-up mode. | • LIN operation mode<br>• LIN wake-up mode | OMM1 bit in RLN3nLMST register | — |
| Successful frame/wake-up transmission | When a frame (header transmission + response transmission), a wake-up signal, or a data group is transmitted successfully. | • When the next communication is started (When the FTS bit in the RLN3nLTRC register is set)<br>• When cleared by software<br>• After transition to LIN reset mode | • LIN operation mode<br>• LIN wake-up mode | FTC flag in RLN3nLST register | √ |
| Successful frame/wake-up reception | When a frame (header transmission + response reception), a wake-up signal, or a data group is received successfully. | • When the next communication is started (When the FTS bit in the RLN3nLTRC register is set)<br>• When cleared by software<br>• After transition to LIN reset mode | • LIN operation mode<br>• LIN wake-up mode | FRC flag in RLN3nLST register | √ |
| Error detection | If any of the RPER flag, CSER flag, FER flag, FTER flag, PBER flag, and BER flag in the RLN3nLEST register is set to 1 (error detected). | • When the next communication is started (When the FTS bit in the RLN3nLTRC register is set)<br>• When cleared by software*1<br>• After transition to LIN reset mode | • LIN operation mode<br>• LIN wake-up mode | ERR flag in RLN3nLST register | √ |
| Successful data 1 reception | When the RFT bit in the RLN3nLDFC register is 0 (reception) and the first byte of the response field or the first byte of each data group is received successfully.*2 | • When the next communication is started (When the FTS bit in the RLN3nLTRC register is set)<br>• When cleared by software<br>• After transition to LIN reset mode | LIN operation mode | D1RC flag in RLN3nLST register | — |
| Successful header transmission | When a header field is transmitted successfully. | • When the next communication is started (When the FTS bit in the RLN3nLTRC register is set)<br>• When cleared by software<br>• After transition to LIN reset mode | LIN operation mode | HTRC flag in RLN3nLST register | √ |

Note 1. In LIN wake-up mode or LIN operation mode, the ERR flag in the RLN3nLST register is cleared to 0 by writing 0 to the RPER flag, CSER flag, FER flag, FTER flag, PBER flag, or BER flag in the RLN3nLEST register.

Note 2. Not detected when the RFDL [3:0] bits in the RLN3nLDFC register are $0000_B$ (0-byte + checksum).

**Table 19.86 Types of Statuses in LIN Slave Mode**

| Status | Status Set Condition | Status Clear Condition | Operating Mode Capable of Status Detection | Corresponding Bit | Interrupt |
|---|---|---|---|---|---|
| Reset | After the OM0 bit in the RLN3nLCUC register is set to not-LIN-reset-mode, if the LIN/UART interface actually exits LIN reset mode. | After the OM0 bit of the RLN3nLCUC register is set to LIN reset mode, if the LIN/UART interface actually enters LIN reset mode. | All modes | OMM0 bit in RLN3nLMST register | — |
| Operation mode | After the OM1 bit in the RLN3nLCUC register is set to LIN operation mode, if the LIN/UART interface actually enters LIN operation mode. | • After the OM1 bit in the RLN3nLCUC register is set to LIN wake-up mode, if the LIN/UART interface actually enters LIN wake-up mode. | • LIN operation mode<br>• LIN wake-up mode | OMM1 bit in RLN3nLMST register | — |
| Successful response/wake-up transmission | When a response field, a wake-up signal, or a data group is transmitted successfully. | • When cleared by software<br>• After transition to LIN reset mode | • LIN operation mode<br>• LIN wake-up mode | FTC flag in RLN3nLST register | √ |
| Successful response/wake-up reception | When a response field, a wake-up signal, or a data group is received successfully. | • When cleared by software<br>• After transition to LIN reset mode | • LIN operation mode<br>• LIN wake-up mode | FRC flag in RLN3nLST register | √ |
| Error detection | If any of the RPER flag, IPER flag, CSER flag, SFER flag, FER flag, TER flag, and BER flag of the RLN3nLEST register is set to 1 (error detected). | • When cleared by software*1<br>• After transition to LIN reset mode | • LIN operation mode<br>• LIN wake-up mode | ERR flag in RLN3nLST register | √ |
| Successful data 1 reception | When the RCDS bit in the RLN3nLDFC register is 0 (reception) and the first byte of the response field or the first byte for each data group is received successfully.*2 | • When cleared by software<br>• After transition to LIN reset mode | LIN operation mode | D1RC flag in RLN3nLST register | — |
| Successful header reception | When a header field is received successfully. | • When cleared by software<br>• After transition to LIN reset mode | LIN operation mode | HTRC flag in RLN3nLST register | √ |

Note 1. In LIN wake-up mode or LIN operation mode, the ERR flag in the RLN3nLST register is cleared to 0 by writing 0 to the RPER flag, IPER flag, CSER flag, SFER flag, FER flag, TER flag, or BER flag of the RLN3nLEST register.

Note 2. Not detected when the RFDL [3:0] bits in the RLN3nLDFC register are $0000_B$ (0-byte + checksum).

## 19.7.7 Error Statuses

### 19.7.7.1 LIN Master Mode

#### (1) Types of Error Statuses

The LIN/UART interface can detect six types of error statuses in LIN master mode. The condition of these error statuses can be checked by means of the corresponding bits in the RLN3nLEST register.

All error statuses represent interrupt events.

**Table 19.87** shows the types of error statuses.

**Table 19.87 Types of Error Statuses in LIN Master Mode**

| Status | Error Detection Condition | Operating Mode Capable of Error Detection | Communication | Enable/ Disable Detection | Corresponding Bit |
|---|---|---|---|---|---|
| Bit error | The transmitted data and the data on the LIN bus monitored by the receive pin do not match *1*2 | • LIN operation mode<br>• LIN wake-up mode | Aborted | Enabled | BER flag in RLN3nLEST register |
| Physical bus error | • LIN bus is detected to be high level when transmitting a break<br>• LIN bus is detected to be low level when transmitting a break delimiter<br>• LIN bus is detected to be high level when transmitting a wake-up | • LIN operation mode<br>• LIN wake-up mode | Aborted | Enabled | PBER flag in RLN3nLEST register |
| Timeout error | A frame or response transmission/reception does not terminate within a given time*3 | LIN operation mode | Aborted | Enabled | FTER flag in RLN3nLEST register |
| Framing error | In response field reception, the stop bit of each data byte is low level | LIN operation mode | Aborted | Enabled | FER flag in RLN3nLEST register |
| Checksum error | In response field reception, checksum test results in an error | LIN operation mode | — | Disabled | CSER flag in RLN3nLEST register |
| Response preparation error | One of the following conditions occurs in frame separate mode during a multi-byte response reception:<br>• The first reception data byte is received after completion of header transmission but before a response transmission/reception request is specified<br>• The first reception data byte is received after the completion of previous data group reception but before a transmission/ reception request for the next data group is specified. | LIN operation mode | Aborted | Disabled | RPER flag in RLN3nLEST register |

Note 1. If a bit error is detected, processing is aborted after a stop bit is sent. If a bit error is detected in a non-data area, such as an inter-byte space, the transmission is aborted immediately after the bit that caused the error is transmitted. If a bit error is detected during the transmission of a wake-up, the transmission of the wake-up is aborted after the bit that caused the error is transmitted.

Note 2. In a multi-byte response transmission, bit errors are also detected between data groups.

Note 3. The timeout time depends on the response field data length (the RFDL [3:0] bits in the RLN3nLDFC register) and the checksum selection (the CSM bit in the RLN3nLDFC register), and can be calculated using the following formula.
When the FSM bit in the RLN3nLDFC register is set to 1 (frame separate mode), the timeout time is that of the 8 data bytes until the RTS bit of the RLN3nLTRC register is set. Once the RTS bit is set, the timeout time is changed to the time based on the response field data length (the RFDL[3:0] bits in the RLN3nLDFC register).

**[Frame timeout]**
When classic checksum is selected (when the CSM bit in RLN3nLDFC is 0): Timeout time = 49 + (number of data bytes + 1) × 14 [Tbit]
When enhanced checksum is selected (when the CSM bit in RLN3nLDFC is 1): Timeout time = 48 + (number of data bytes + 1) × 14 [Tbit]
The aforementioned timeout time is longer than the TFRAME_MAX of LIN Specification Package Revision 1.3 when classic checksum is selected, or the TFRAME_MAX of LIN Specification Package Revision 2.x when enhanced checksum is selected.
**[Response timeout]**
Timeout time = (number of data bytes + 1) × 14 [Tbit]

When an error is detected, time-out error detection function stops.

The error status is cleared when the next communication is started (when the FTS bit in the RLN3nLTRC register is set), when it is cleared by software, or at a transition to LIN reset mode.

### (2) Target Time Domain for LIN Error Detection

**Figure 19.19** shows the time domain in which the LIN/UART interface in LIN master mode performs monitoring for error detection.

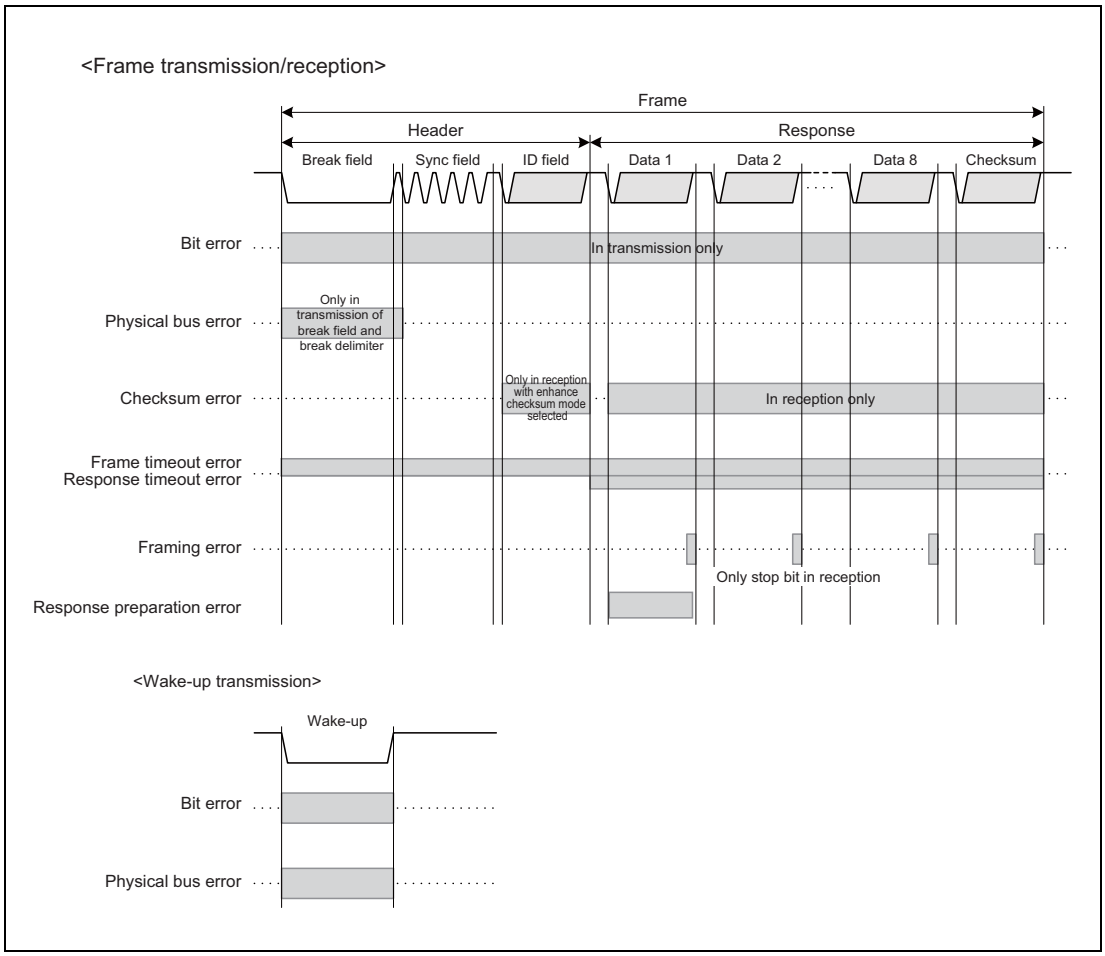



**Figure 19.19 Target Time Domain for LIN Error Detection (LIN Master Mode)**

### 19.7.7.2 LIN Slave Mode

#### (1) Types of Error Statuses

The LIN/UART interface can detect seven types of error statuses in LIN slave mode [auto baud rate] or in LIN slave mode [fixed baud rate]. These error statuses can be verified by checking the corresponding bits in the RLN3nLEST register.

**Table 19.88** shows the types of error statuses.

**Table 19.88 Types of Error Statuses in LIN Slave Mode**

| Status | Error Detection Condition | Operating Mode Capable of Error Detection | Communication | Enable/ Disable Detection | Corresponding Bit |
|---|---|---|---|---|---|
| Bit error | The transmitted data and the data on the LIN bus monitored by the receive pin do not match*1 *2 | • LIN operation mode<br>• LIN wake-up mode | Aborted | Enabled | BER flag in RLN3nLEST register |
| Timeout error | A frame or response transmission/reception does not terminate within a given time*3 | LIN operation mode | Aborted | Enabled | TER flag in RLN3nLEST register |
| Framing error | In frame reception, the stop bit of each data byte is low level | LIN operation mode | Aborted | Enabled | FER flag in RLN3nLEST register |
| Sync field error | If the width of the break low level is greater than the width set by the LBLT bit in the RLN3nLBFC register and the sync field is not $55_H$ | LIN operation mode | Aborted | Enabled*4 | SFER flag in RLN3nLEST register |
| Checksum error | In response field reception, the checksum test results in an error | LIN operation mode | —*5 | Disabled | CSER flag in RLN3nLEST register |
| ID parity error | If the received ID parity bit does not match the value that is automatically calculated by the LIN/ UART interface | LIN operation mode | Aborted | Enabled | IPER flag in RLN3nLEST register |
| Response preparation error | • After the reception of a header, if the response is not prepared before the first reception data byte is received<br>• In a multi-byte response reception, if the preparation for the reception of next data group does not complete before the first reception data byte for the next data group is received | LIN operation mode | Aborted | Disabled | RPER flag in RLN3nLEST register |

Note 1. If a bit error is detected, processing is aborted after a stop bit is sent. If a bit error is detected in a non-data area, such as an inter-byte space, the transmission is aborted immediately after the bit that caused the error is transmitted. If a bit error is detected during the transmission of a wake-up, the transmission of the wake-up is aborted after the bit that caused the error is transmitted.

Note 2. In a multi-byte response transmission, bit errors are also detected between data groups.

Note 3. The timeout time depends on the response field data length (the RFDL [3:0] bits in the RLN3nLDFC register) and the checksum selection (the LCS bit in the RLN3nLDFC register), and this can be calculated according to the following formula. The timeout time is that of 8 data bytes until the RTS bit or the LNRR bit of the RLN3nLTRC register is set. When the RTS bit is set, the timeout time is changed to the time based on the response field data length (RFDL[3:0] bit of the RLN3nLDFC register). When the LNRR bit is set, the timeout function stops.

**[Frame timeout]**
When classic checksum is selected (when the CSM bit in RLN3nLDFC is 0): Timeout time = 49 + (number of data bytes + 1) × 14 [Tbit]
When enhanced checksum is selected (when the CSM bit in RLN3nLDFC is 1): Timeout time = 48 + (number of data bytes + 1) × 14 [Tbit]

The aforementioned timeout time is longer than the TFRAME_MAX of LIN Specification Package Revision 1.3 when classic checksum is selected, or the TFRAME_MAX of LIN Specification Package Revision 2.x when enhanced checksum is selected.

**[Response timeout]**
Timeout time = (number of data bytes + 1) × 14 [Tbit]
When en error is detected, time-out error detection function stops.

Note 4. Only reflection of the result to the SFER flag can be enabled/disabled. Error detection cannot be enabled/disabled.

Note 5. Checksum determination is performed upon completion of response frame reception. In case of an error, the successful reception flag is not set to 1.

The error status is cleared by software or at a transition to LIN reset mode.

### (2) Target Time Domain for LIN Error Detection

**Figure 19.20** shows the time domain in which the LIN/UART interface in slave mode performs monitoring for error detection.

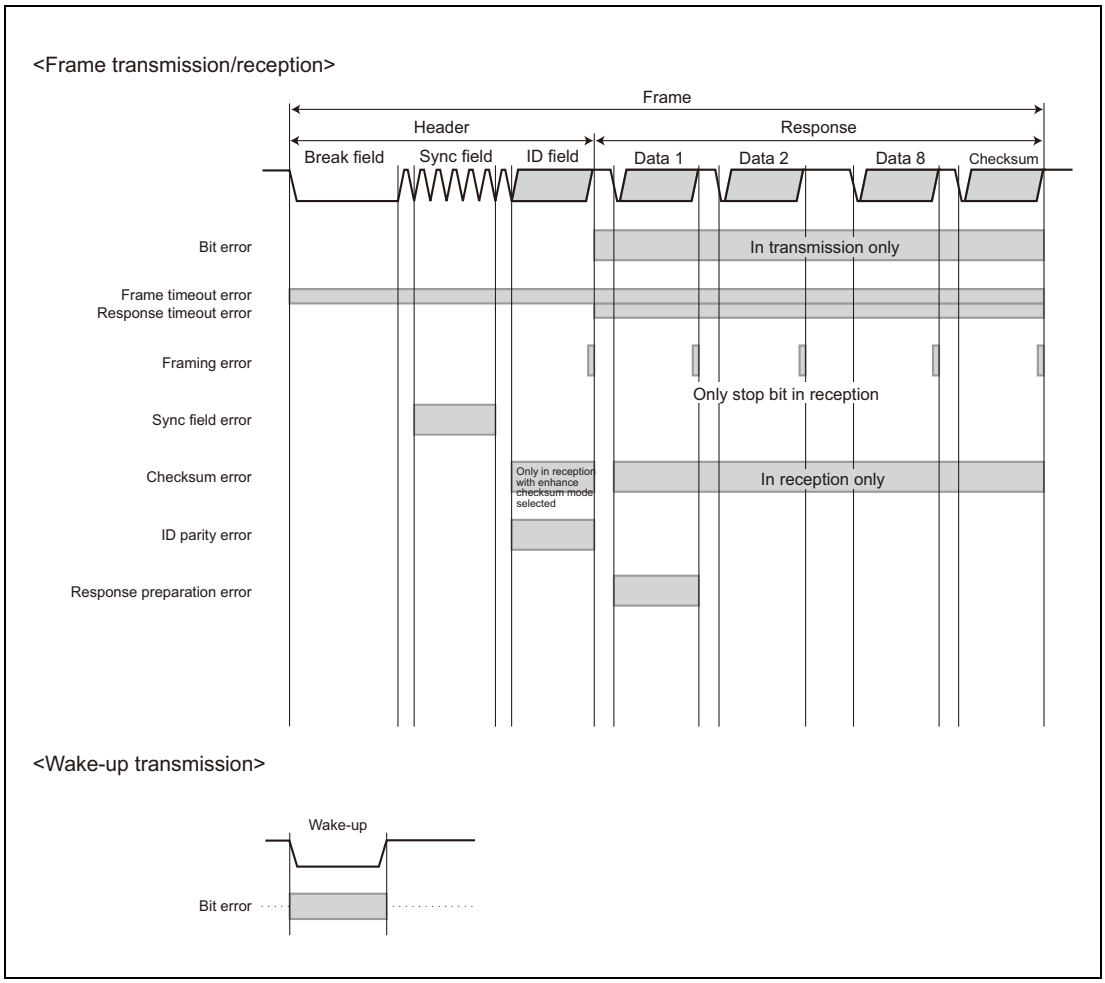



**Figure 19.20 Target Time Domain for LIN Error Detection (LIN Slave Mode)**

## 19.8 UART Mode

In LIN reset mode, setting the LMD bits in the RLN3nLMD register to $01_B$ (UART mode) and the OM0 bit in the RLN3nLCUC register to 1 changes the mode to UART mode, turning the OMM0 bit in the RLN3nLMST register to 1.

### 19.8.1 Transmission

**Figure 19.21** shows the operation of the LIN/UART interface (in UART mode) during transmission. **Table 19.89** provides processing of the LIN/UART interface (in UART mode) during transmission.

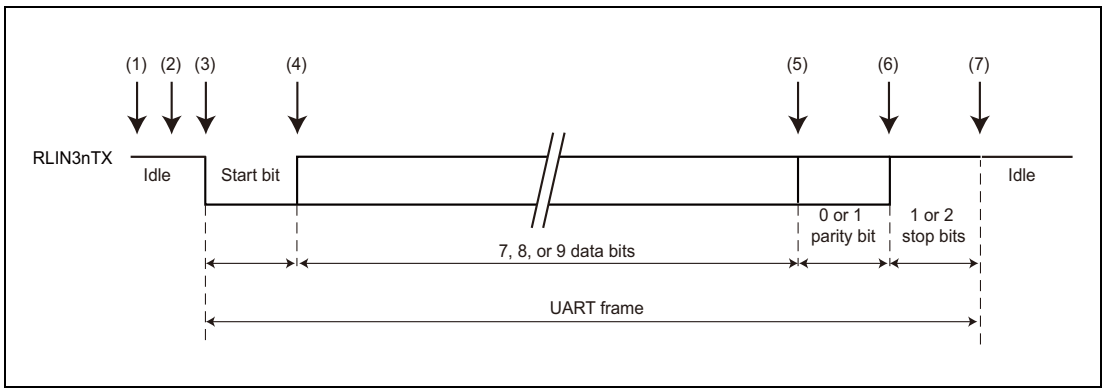



**Figure 19.21 LIN/UART Interface (in UART Mode) Transmission Operation**

**Table 19.89 LIN/UART Interface (UART Mode) Transmission Processing (1/2)**

| | Software Processing | LIN/UART Interface Processing |
|---|---|---|
| (1) | • Sets a baud rate.<br>• Sets noise filter ON/OFF.<br>• Enables error detection.<br>• Configures data format.<br>• Sets an interrupt generation timing.<br>• Exits from LIN reset mode.<br>• Sets the transmission enable bit (UTOE bit) to 1. | • Waits for a transmission trigger (RLN3nLUTDR register) by software. |
| (2) | • Specifies the data to be transmitted in the UART transmit data register (RLN3nLUTDR) or UART wait transmit data register (RLN3nLUWTDR). | • Sets the transmission status flag. |
| (3) | • Waits for an interrupt request.<br>[When the UTIGTS bit is 0 (a transmission interrupt request is generated upon start of transmission)] | • Transmits a start bit (for switching from reception to transmission in half duplex communication, transmits a start bit after receiving 1 stop bit. For details about this function, see **Section 19.8.1.4, Transmission Start Wait Function**.<br>[When the UTIGTS bit is 0 (a transmission interrupt request is generated upon start of transmission)]<br>• Outputs a transmission interrupt. |
| (4) | • When transmitting data continuously, specifies the next data to be transmitted in the UART transmission data register (RLN3nLUTDR) and waits for the generation of an interrupt request. | Transmits the data specified in the UART (wait) transmission data register. |
| (5) | | Transmits a parity bit when parity is used. |
| (6) | | Transmits 1 or 2 stop bits. |

**Table 19.89 LIN/UART Interface (UART Mode) Transmission Processing (2/2)**

| | Software Processing | LIN/UART Interface Processing |
|---|---|---|
| (7) | [When the UTIGTS bit is 0 (a transmission interrupt request is generated upon start of transmission)]<br>• If another item of transmission data is set, goes to step (3). | [When the UTIGTS bit is 0 (a transmission interrupt request is generated upon start of transmission)]<br>• If another piece of transmission data is set, goes to step (3).<br>• If the next data to be transmitted is not specified, clears the transmission status flag. |
| | [When the UTIGTS bit is 1 (a transmission interrupt is output upon completion of transmission)]<br>• When transmitting data continuously, goes to step (2). | [When the UTIGTS bit is 1 (a transmission interrupt is output upon completion of transmission)]<br>• Generates RLIN3n transmission interrupt request.<br>• Clears the transmission status flag. |

### 19.8.1.1 Continuous Transmission

The LIN/UART interface (in UART mode) can transmit multiple sets of data continuously by using the RLN3nLUTDR register. **Figure 19.22** shows an operation example where the transmission interrupt generation timing is the start of transmission and an operation example where the transmission interrupt generation timing is the completion of transmission.

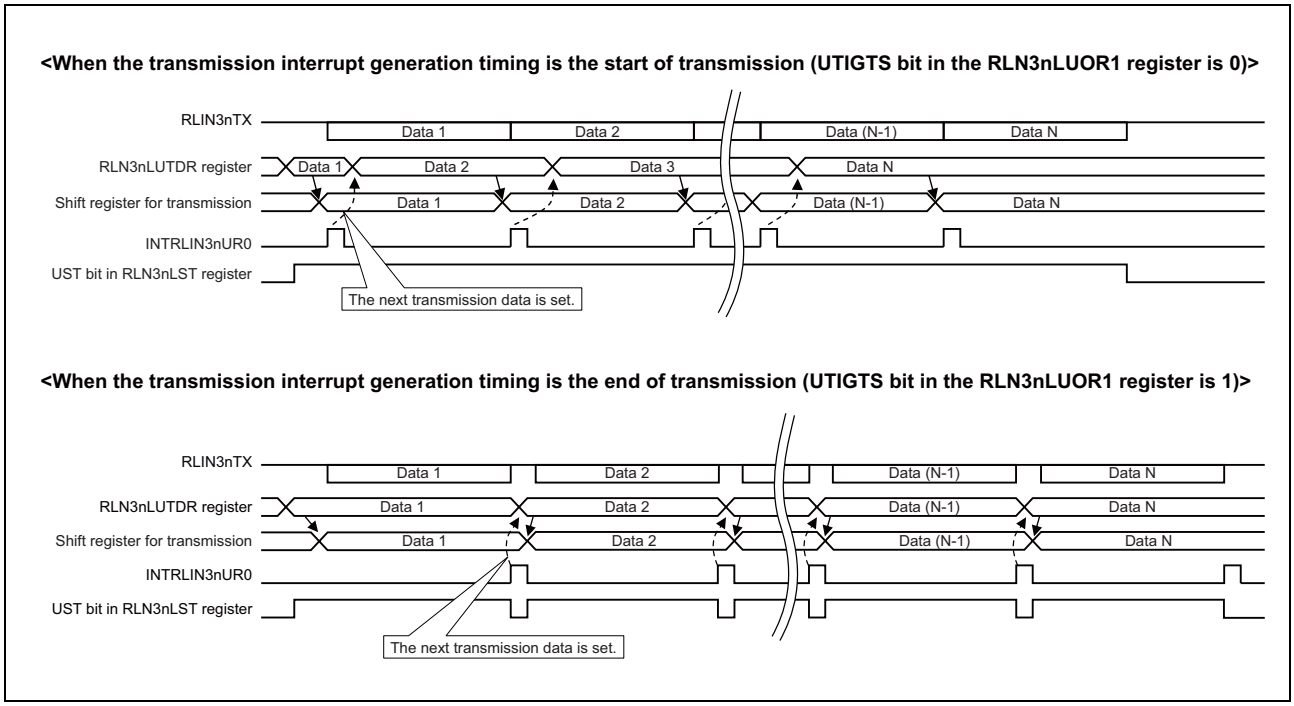



**Figure 19.22 Operation Examples of LIN/UART Interface (UART Mode) Continuous Transmission**

An interrupt can be generated at the completion of a transmission by changing the UTIGTS bit in the RLN3nLUOR1 register from 0 to 1 after the start of transmission of final data, provided that the transmission interrupt generation timing is the start of transmission and the completion of transmission of final data needs to be known.

### 19.8.1.2 UART Buffer Transmission

The LIN/UART interface (in UART mode) has a maximum of nine bytes of UART buffers and it is capable of performing continuous transmissions through the use of UART buffers.

**Figure 19.23** shows the UART buffer transmission operation of the LIN/UART interface (in UART mode). **Table 19.90** shows the UART buffer transmission processing.

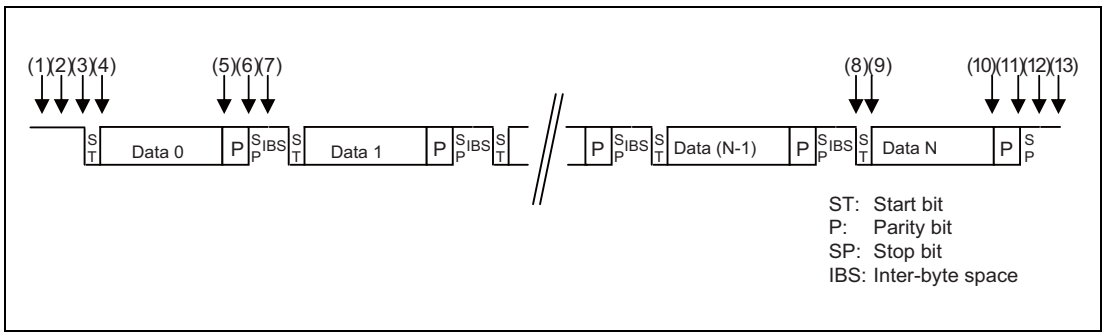



**Figure 19.23 UART Buffer Transmission in LIN/UART Interface (in UART Mode)**

**Table 19.90 UART Buffer Transmission Processing in LIN/UART Interface (in UART Mode) (1/2)**

| | Software Processing | LIN/UART Interface Processing |
|---|---|---|
| (1) | • Sets a baud rate<br>• Sets noise filter ON/OFF<br>• Enables error detection<br>• Configures data format<br>• Sets the interrupt generation timing to the completion of transmission.<br>• Exits from LIN reset mode.<br>• Sets the transmission enable bit (UTOE bit) to 1 | • Waits for a transmission trigger (RTS bit) by software |
| (2) | • Sets the UART buffer data length and whether the start of transmission must be waited.<br>• Specifies the data to be transmitted in the UART data 0 buffer register (RLN3nLUDB0) and the LIN data buffer b register (RLN3nLDBRb). (b =1 to 8)<br>• Sets the UART buffer transmission start bit (RTS). | • Sets the transmission status flag. |
| (3) | Waits for an interrupt request. | Transmits a start bit.<br>(For switching from reception to transmission in half duplex communication, transmits a start bit after receiving 1 stop bit. For details about this function, see **Section 19.8.1.4, Transmission Start Wait Function**.) |
| (4) | | Transmits the data specified in the UART data buffer 0 register (RLN3nLUDB0) or the LIN/UART data buffer b register (RLN3nLDBRb). |
| (5) | | Transmits a parity bit when parity is used. |
| (6) | | Transmits 1 or 2 stop bits<br>(When the number of data set in UART buffer data length select bits is 1, proceeds to (12).) |
| (7) | | Transmits an inter-byte space (idle). |
| | | Repeats steps (3) to (7) until the number of data set in the UART buffer data length select bits -1 is reached. |

**Table 19.90 UART Buffer Transmission Processing in LIN/UART Interface (in UART Mode) (2/2)**

| | Software Processing | LIN/UART Interface Processing |
|---|---|---|
| (8) | | Transmits a start bit. |
| (9) | | Transmits the data specified in the LIN/UART data buffer b register (RLN3nLDBRb). |
| (10) | | Transmits a parity bit when parity is used. |
| (11) | | Transmits 1 or 2 stop bits. |
| (12) | | • Sets the successful buffer transmission flag.<br>• Clears the UART buffer transmission start bit (RTS).<br>• Generates a transmission interrupt request signal.<br>• Clears the transmission status flag. |
| (13) | • Checks the RLN3nLST register, and clears flags<br>• When continuously transmitting data, goes to step (2). | |

### (1) UART Buffer Transmission

For a 9-byte transmission, the contents stored in the RLN3nLUDB0 and RLN3nLDBR1 to RLN3nLDBR8 registers are transmitted to data regions 0 to 8. The RLN3nLUDB0 register is used only if 9-byte transmission is specified. In other cases, the RLN3nLDBR1 to RLN3nLDBR8 registers are selected depending on the data length. For a 4-byte transmission, the contents stored in the RLN3nLDBR1 to RLN3nLDBR4 registers are transmitted to data regions 1 to 4, but the contents of the RLN3nLDBR5 to RLN3nLDBR8 registers are not transmitted. An RLIN3n transmission interrupt is generated after the number of data specified in the MDL [3:0] bits of the RLN3nLDFC register is transmitted. The spaces between each transmitted data can be specified in the IBS bit of the RLN3nLSC register.

**Figure 19.24** shows a 9-byte UART buffer and the transmission processing.

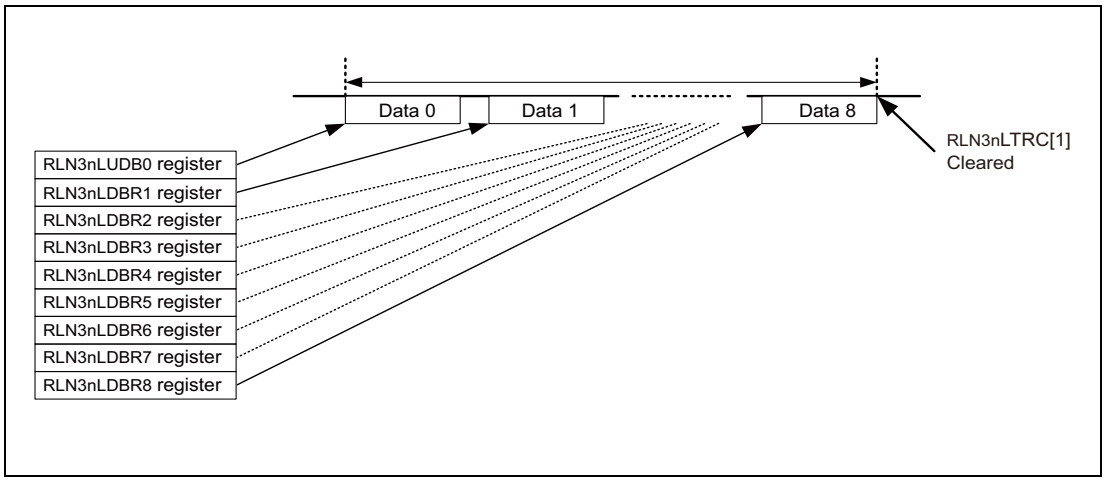



**Figure 19.24 UART Buffer and Transmission Processing (for 9-Byte Transmission)**

### 19.8.1.3 Data Transmission

One bit of data is transmitted per Tbit.

In half-duplex communication, if the BERE bit in the RLN3nLEDE register is 1 (bit error detection enabled), the transmission data and the input pin level are compared bit by bit during data transmission, and the results are stored in the BER flag of the RLN3nLEST register (see **Section 19.7.7, Error Statuses**). The timing at which the input pin is sampled during data transmission can vary depending on the settings of the LPRS[2:0] and NSPB[3:0] bits in the RLN3nLWBR register.

The bit error detection timing in UART mode is shown in **Table 19.91**.

**Table 19.91 Error Detection Timing in UART Mode**

| Sampling Count Per Bit | Bit Error Detection Timing |
|---|---|
| 6 samples | 3rd clock cycle + 1 prescaler clock |
| 7 samples | 4th clock cycle + 1 prescaler clock |
| 8 samples | 4th clock cycle + 1 prescaler clock |
| 9 samples | 5th clock cycle + 1 prescaler clock |
| 10 samples | 5th clock cycle + 1 prescaler clock |
| 11 samples | 6th clock cycle + 1 prescaler clock |
| 12 samples | 6th clock cycle + 1 prescaler clock |
| 13 samples | 7th clock cycle + 1 prescaler clock |
| 14 samples | 7th clock cycle + 1 prescaler clock |
| 15 samples | 8th clock cycle + 1 prescaler clock |
| 16 samples | 8th clock cycle + 1 prescaler clock |

Example of Data Transmission Timing (when 1 Tbit = 16 samplings) is shown in **Figure 19.25**

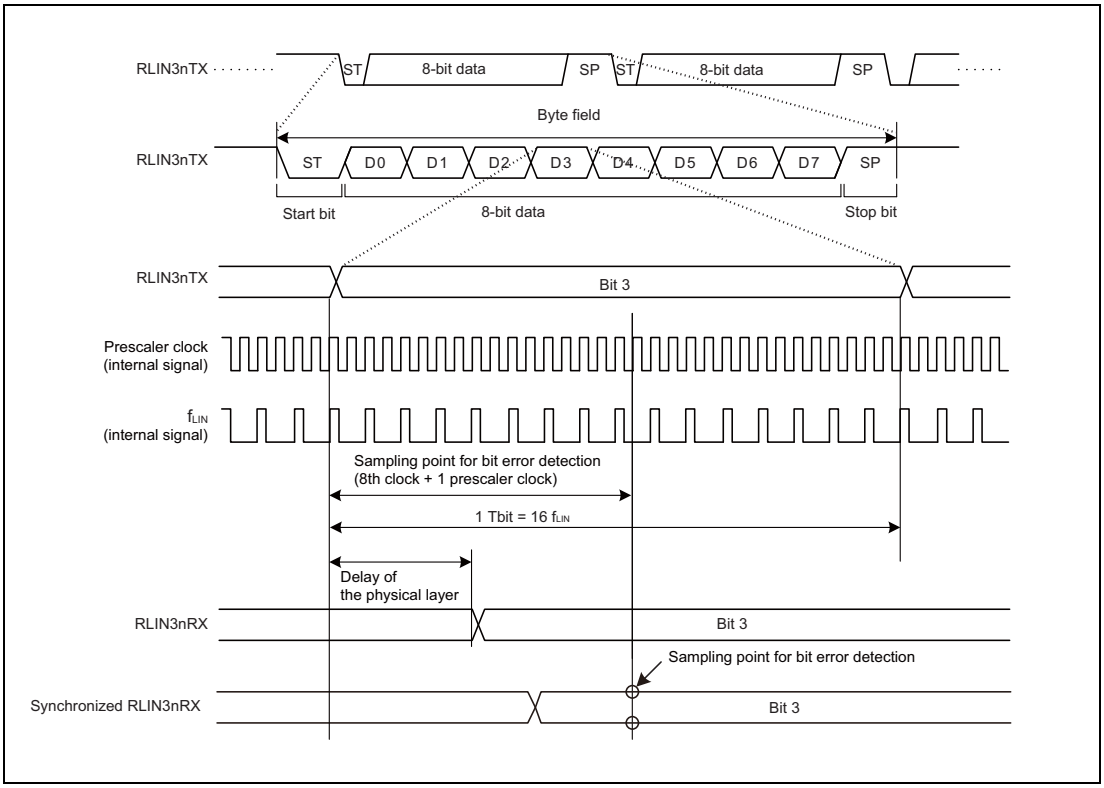



**Figure 19.25 Example of Data Transmission Timing (When 1 Tbit = 16 samplings)**

### 19.8.1.4 Transmission Start Wait Function

For performing half-duplex communication, the LIN/UART interface (in UART mode) has the function of securing the reception stop bit length when switching from reception to transmission.

If it is desired to delay the start of transmission until the stop bits for the reception are completed, specify the transmission data in the RLN3nLUWTDR register, which is used only for the wait function, instead of specifying it in the RLN3nLUTDR register as a transmission start request. When transmitting from the UART buffer, set the RTS bit in the RLN3nLTRC register to 1 (UART buffer transmission start) while the UTSW bit in the RLN3nLDFC register is set to 1.

In this case, the LIN/UART interface delays the start of transmission until the stop bits of reception data are completed.

It should be noted that even if the UART stop bit length select bit (USBLS) in RLN3nLBFC register is 1 (stop bits = 2 bits), there is only a 1-bit delay.

**Figure 19.26** shows the operation of transmission wait function.

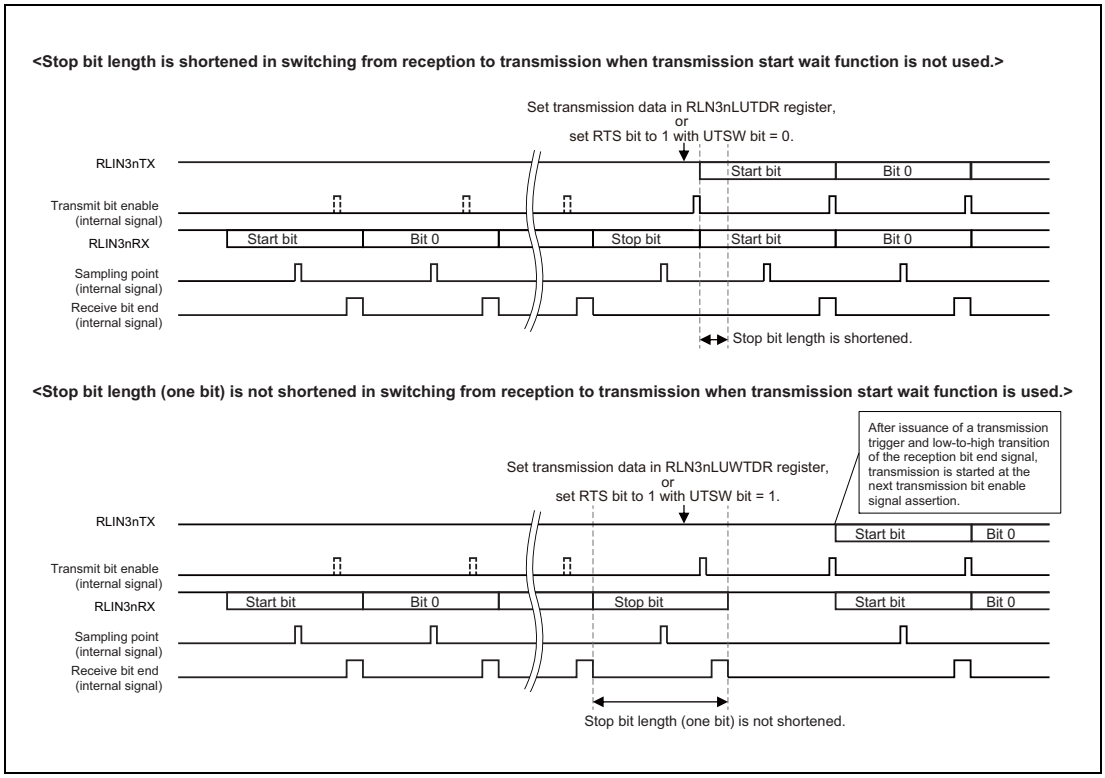



**Figure 19.26 Transmission Wait Function (Case When Transmit Data Is Set While Stop Bits Are Being Received)**

### 19.8.2 Reception

**Figure 19.27** shows the LIN/UART interface (in UART mode) reception operation.**Table 19.92** shows the LIN/UART interface (in UART mode) reception processing.

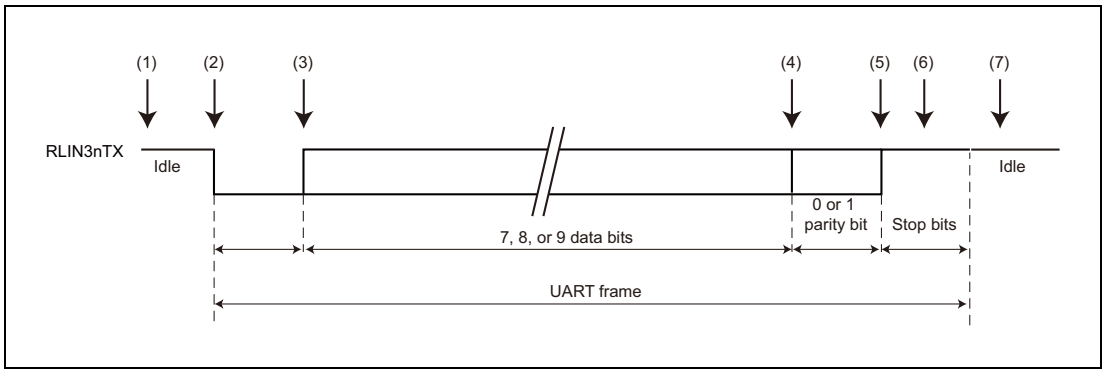



**Figure 19.27 LIN/UART Interface (in UART Mode) Reception Operation**

**Table 19.92 LIN/UART Interface (in UART Mode) Reception Processing**

| | Software Processing | LIN/UART Interface Processing |
|---|---|---|
| (1) | • Sets a baud rate.<br>• Sets noise filter ON/OFF.<br>• Enables error detection.<br>• Configures data format.<br>• Exits from LIN reset mode.<br>• Sets the reception enable bit (UROE bit) to 1. | • Waits for the reception to be enabled by software.<br>• Waits for detection of a start bit. |
| (2) | Waits for an interrupt request. | • Waits for a falling edge from the reception pin, and detects a start bit.<br>• Sets the reception status flag. |
| (3) | | Receives data. |
| (4) | | Receives a parity bit when parity is used. |
| (5) | | Receives only 1 stop bit. |
| (6) | | • Generates a RLIN3n successful reception interrupt request.<br>• Clears the reception status flag. |
| (7) | Checks the RLN3nLST register, and clears flags | Waits for a falling edge from the reception pin. |

### 19.8.2.1 Data Reception

Data reception is performed by using the synchronized RLIN3nRX (an internal signal) that is the input from the RLIN3nRX pin synchronized with the prescaler clock.

The byte field is synchronized at the falling edge of the start bit for the synchronized RLIN3nRX signal. After the falling edge is detected, resampling is performed 0.5 Tbits later when the number of sampling per 1 Tbit is even and {(the number of sampling + 1) / 2} / (the number of sampling) Tbits later when the number is odd. If the synchronized RLIN3nRX signal is low level, the bit is recognized as a start bit. The bit is not recognized as a start bit if the RLIN3nRX signal after the reset is de-asserted is fixed to low level or if a high level is detected during the resampling.

After the start bit is detected, 1 bit is sampled per Tbit.

However, when the BERE bit in the RLN3nLEDE register is 1, the sampling point is the same as the bit error detection timing.

The LIN/UART interface has a noise filter function for received data. If the LRDNFS bit in the RLN3nLMD register is 0, the noise filter is used. For a sampling value, the value determined by a 3-sampling majority rule by the prescaler clock is used. If the LRDNFS bit in the RLN3nLMD register is 1, the noise filter is not used. In this case, for a sampling value, the synchronized RLIN3nRX value at the sampling position is used as is.

**Figure 19.28** shows an example of data reception timing.

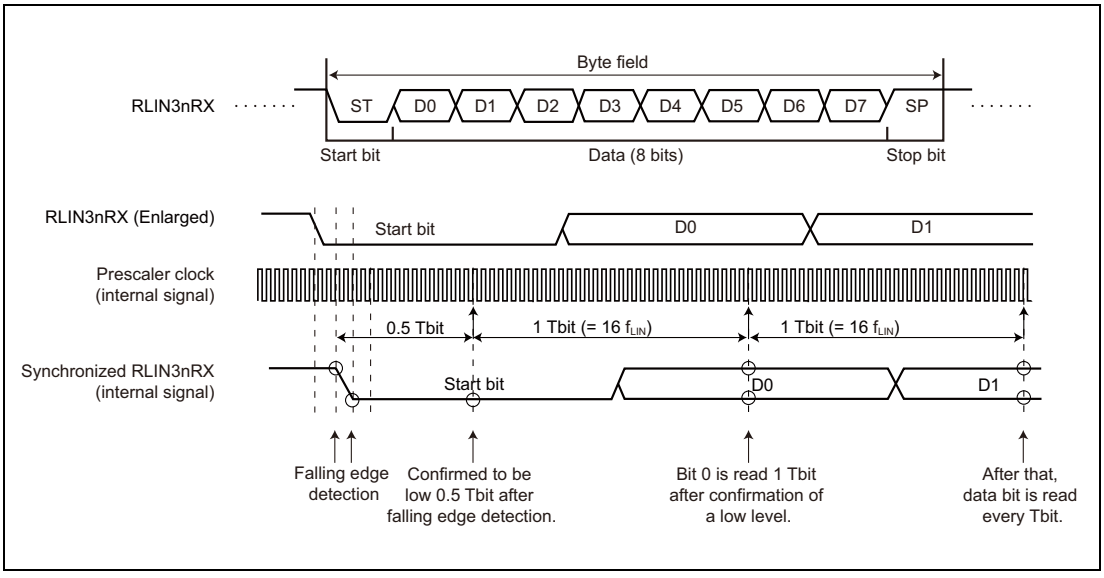



**Figure 19.28 Example of Data Reception Timing (When 1 Tbit = 16 samplings)**

## 19.8.3 Expansion Bits

The LIN/UART interface (in UART mode) can transmit and receive 9-bit long data by setting the UEBE bit in the RLN3nLUOR1 register to 1.

### 19.8.3.1 Expansion Bit Transmission

The LIN/UART interface (in UART mode) can transmit 9-bit long data by writing the 9-bit data to either the UART transmission data register (RLN3nLUTDR) or the UART wait transmission data register (RLN3nLUWTDR) when the expansion bit enable bit (UEBE) in the UART option register 1 (RLN3nLUOR1) is 1.

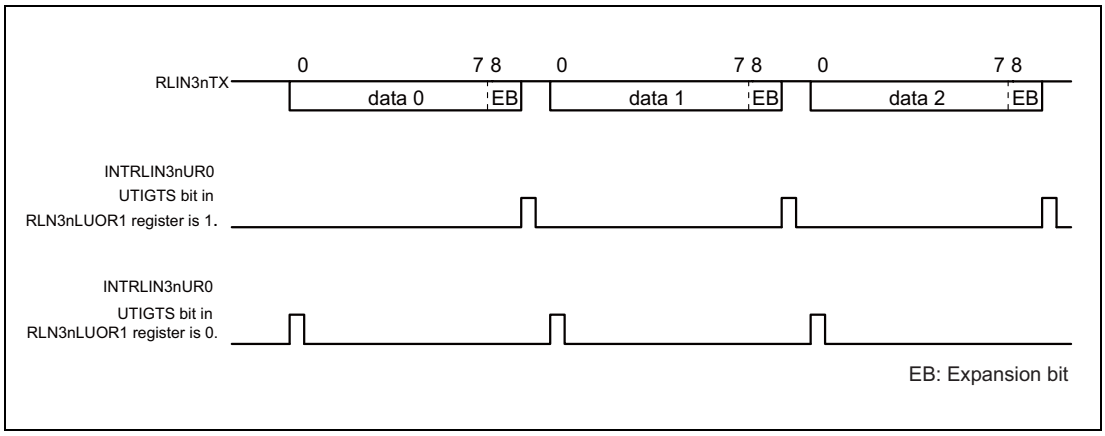



**Figure 19.29 Transmission Example When Expansion Bit is Enabled (LSB First)**

### 19.8.3.2 Expansion Bit Reception

The LIN/UART interface (in UART mode) can always receive 9-bit data without comparing the expansion bits when the expansion bit enable bit (UEBE) in the UART option register 1 (RLN3nLUOR1) is 1, the expansion bit comparison disable bit (UECD) is 1, and the expansion bit data comparison enable bit (UEBDCE) is 0. Regardless of the expansion bit detection level select bit (UEBDL) setting in the UART option register 1 (RLN3nLUOR1), a RLIN3n successful reception interrupt is generated when 9-bit data is received.

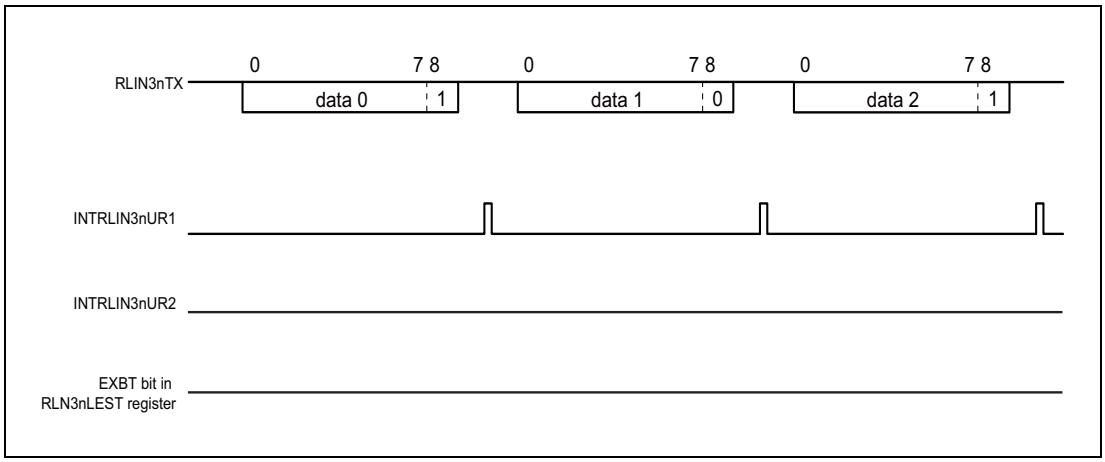



**Figure 19.30 Expansion Bit Reception Example (LSB First)**

### 19.8.3.3 Expansion Bit Reception (with Expansion Bit Comparison)

The LIN/UART interface (in UART mode) can compare received expansion bits and the UEBDL bits when the expansion bit enable bit (UEBE) in the UART option register 1 (RLN3nLUOR1) is 1 and the expansion bit comparison disable bit (UECD) is 0 and the expansion bit/data comparison enable bit (UEBDCE) is 0.

If the level that was set in the expansion bit detection level select bit (UEBDL) is detected, an RLIN3n status interrupt request is generated upon completion of data reception, and the expansion bit detection flag (EXBT) in the LIN error status register (RLN3nLEST) is set. If the reversed value of an expansion bit detection level is detected, RLIN3n successful reception interrupt request is generated. In either case, the received data is stored in the UART reception data register (RLN3nLURDR), unless there was an overrun error.

**Figure 19.31** shows an example when the expansion bit detection level select bit (UEBDL) is set to 0.

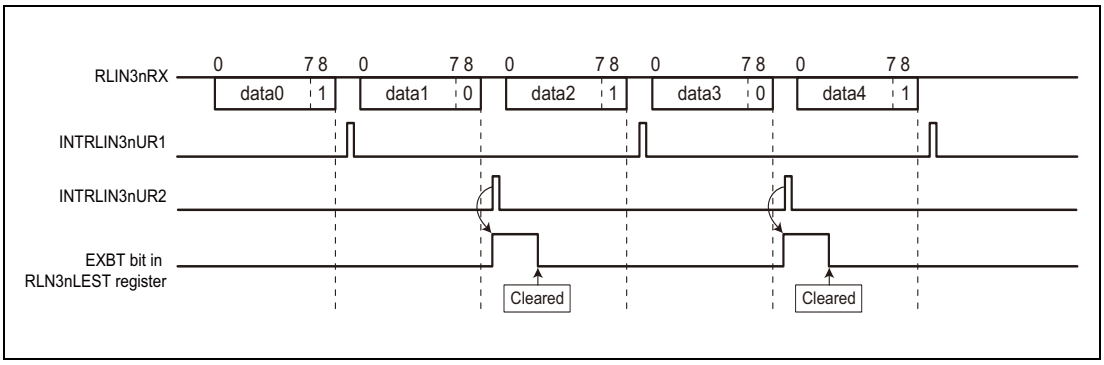



**Figure 19.31 Expansion Bit Reception Example (with Expansion Bit Comparison) (LSB First, UEBDL = 0)**

**NOTE**

- If a reception error (parity error, framing error, or overrun error) occurs in received data 0, 2, or 4 (if a reversed value of an expansion bit detection level is detected), an RLIN3n status interrupt is generated, and the error flag is updated. In this case, a RLIN3n successful reception interrupt is not generated.
- If a reception error (parity error, framing error, or overrun error) occurs in received data 1 or 3 (if an expansion bit detection level is detected), an RLIN3n status interrupt is generated, and the error flag is updated. If the overrun error occurs, the expansion bit detection flag (EXBT) is also set.

### 19.8.3.4 Expansion Bit Reception (with Data Comparison)

If the expansion bit enable bit (UEBE) in the UART option register 1 (RLN3nLUOR1) is 1 and the expansion bit comparison disable bit (UECD) is 0 and the expansion bit/data comparison enable bit (UEBDCE) is 1, when the level that was set by the expansion bit detection level select bit (UEBDL) is detected, the LIN/UART interface (in UART mode) compares the 8 bits of the received data excluding the expansion bit, with the a preset RLN3nLIDB register value.

If the result of the comparison is a match, the LIN/UART interface performs the following operations:

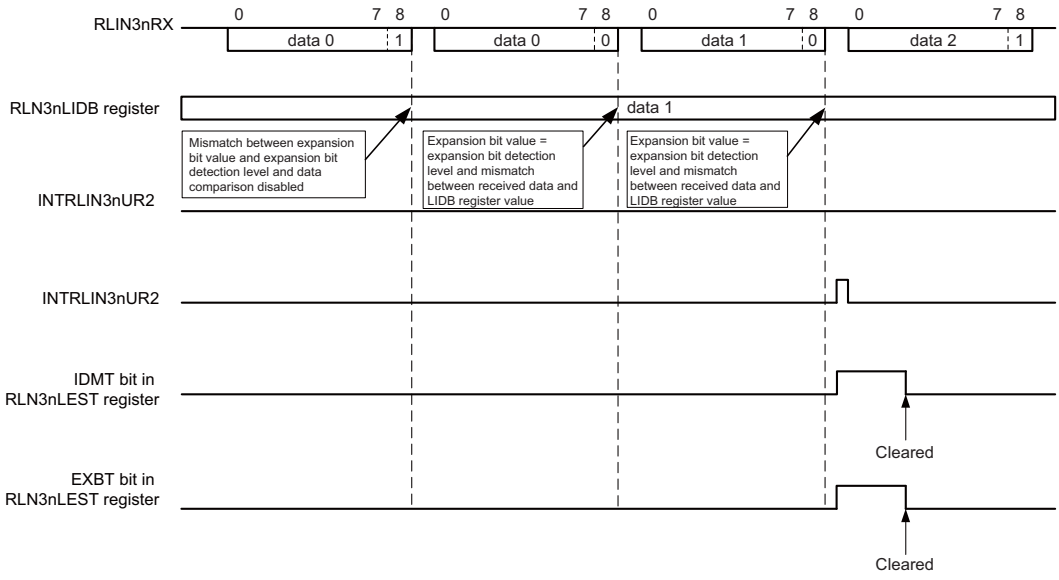
- Generates an RLIN3n status interrupt
- Sets the expansion bit detection flag (EXBT)
- Sets the ID match flag (IDMT)
- Stores the received data in the UART reception data register (RLN3nLURDR)

Even when the result of the comparison is a match, successful RLIN3n successful reception interrupt is not generated.

If the result of the comparison is not a match, no RLIN3n successful reception interrupt or RLIN3n status interrupt is generated, and the EXBT and IDMT flags are not set to 1. The received data is not stored in the UART reception data register (RLN3nLURDR).

When changing the UEBDCE bit to 0, make the change before the reception of the next data is completed.

**Figure 19.32** shows an example when the expansion bit detection level select bit (UEBDL) is set to 0.




**Note:** If a reception error (parity error, framing error, or overrun error) occurs, an RLIN3n status interrupt is generated, and the error flag is updated. If the overrun error occurs and the result of comparison is a match, the EXBT and IDMT are also set to 1.

**Figure 19.32 Expansion Bit Reception Example (with Data Comparison) (LSB First, UEBDL = 0)**

### 19.8.4 Statuses

In UART mode, the LIN/UART interface can detect five types of statuses.

Two statuses, successful UART buffer transmission and error detection, can generate interrupt requests.

**Table 19.93** shows the types of statuses available in UART mode.

**Table 19.93 Types of Statuses in UART Mode**

| Status | Status Set Condition | Status Clear Condition | Corresponding Bit | Interrupt |
|---|---|---|---|---|
| Reset | After the OM0 bit in the RLN3nLCUC register is set to not-LIN-reset-mode, if the LIN/UART interface actually exits LIN reset mode. | After the OM0 bit in the RLN3nLCUC register is set to LIN reset mode, if the LIN/UART interface actually enters LIN reset mode. | OMM0 bit in RLN3nLMST register | — |
| Successful UART buffer transmission | • When the UTIGTS bit in the RLN3nLUOR1 register is 0 (transmission interrupt request is generated upon start of transmission), if the transmission of the last data (the data length is specified by the MDL bits in the RLN3nLDFC register) is started.<br>• When the UTIGTS bit in the RLN3nLUOR1 register is 1 (transmission interrupt request is generated upon completion of transmission), if the transmission of the data length specified by the MDL bit in the RLN3nLDFC register is completed. | • When cleared by software<br>• After transition to LIN reset mode | FTC flag in RLN3nLST register | √ |
| Error detection | If any of the UPER flag, IDMT flag, EXBT flag, FER flag, OER flag, and BER flag in the RLN3nLEST register is set to 1 (error detected). | • When cleared by software*1<br>• After transition to LIN reset mode | ERR flag in RLN3nLST register | √ |
| Transmission status | • When data is written to the RLN3nLUTDR or RLN3nLUWTDR register.<br>• When 1 is written to the RTS bit in the RLN3nLTRC register. | • When the transmission of the data specified in the RLN3nLUTDR or RLN3nLUWTDR register is completed, but the next transmission data is not specified.<br>• When the transmission of the data in the UART buffer is completed and the RTS bit in the RLN3nLTRC register is cleared.<br>• After transition to LIN reset mode | UTS flag in RLN3nLST register | — |
| Reception status | • When a start bit is detected. | • When a sampling point for stop bits is detected<br>• After transition to LIN reset mode | URS flag in RLN3nLST register | — |

Note 1. Writing a 0 to the UPER, IDMT, EXBT, FER, OER, and BER flags in the RLN3nLEST register when not in the LIN reset mode sets the ERR flag in the RLN3nLST register to 0.

## 19.8.5 Error Statuses

### Types of Error Statuses

In UART mode, the LIN/UART interface can detect four types of errors and two types of statuses. The condition of these statuses can be checked by means of the corresponding bits in the RLN3nLEST register.

**Table 19.94** lists applicable status types.

**Table 19.94 Types of Statuses in UART Mode**

| Status | Error Detection Condition | Communication | Enable/ Disable Detection | Corresponding Bit |
|---|---|---|---|---|
| Bit error | The transmitted data and the data monitored on the reception pin do not match*1 | Continues until the transmission of the specified transmission data is completed. | Enabled | BER flag in RLN3nLEST register |
| Overrun error | After received data is stored in the RLN3nLURDR register, the next data is received before the previous data is read. (In this case, no data is stored in the RLN3nLURDR register). | — (Reception is completed by the time this error is detected) | Enabled | OER flag in RLN3nLEST register |
| Framing error | When the first stop bit is low level in the reception processing. | — (Reception is completed by the time this error is detected) | Enabled | FER flag in RLN3nLEST register |
| Parity error | The received parity value fails to match the parity value calculated from the received data | Continues until the data reception is completed. | Disabled *2 | UPER flag in RLN3nLEST register |
| Expansion bit detection | The value of the received expansion bit matches the value of the UEBDL bit in the RLN3nLUOR1 register. | — | Enabled | EXBT flag in RLN3nLEST register |
| ID match detection | The value of the received expansion bit matches the value of the UEBDL bit in the RLN3nLUOR1 register and the 8-bit received data excluding the expansion bit matches the value of the RLN3nLIDB register. | — | Enabled | IDMT flag in RLN3nLEST register |

Note 1. In the case of transmission from the UART buffer, bit errors are detected even in the space between UART frames (inter-byte space).

Note 2. Setting the UPS[1:0] bits in the RLN3nLBFC register to $10_B$ (0 parity) disables the checking of parity bit values. In this case, no parity error is generated.

The error status is cleared by software or at a transition to LIN reset mode.

## 19.9 LIN Self-Test Mode

The LIN/UART interface provides a LIN self-test mode. When the LIN/UART interface enters the LIN self-test mode, RLIN3nTX and RLIN3nRX are disconnected from external pins and they are internally connected within the LIN/UART interface. Therefore, the frame transmitted from RLIN3nTX is looped back to RLIN3nRX. The LIN self-test mode can perform tests exclusively in LIN mode.

The self-test can be performed in the following four modes:

- LIN master self-test mode (transmission): Header transmission and response transmission
- LIN master self-test mode (reception): Header transmission and response reception
- LIN slave self-test mode (transmission): Header reception and response transmission
- LIN slave self-test mode (reception): Header reception and response reception

In LIN self-test mode, the operation is performed at the fastest baud rate, regardless of the setting of the baud rate generator.

Regardless of the setting of the baud rate related registers, the baud rate setting is the LIN communication clock source/16 [bps]. (The NSPB bits in the RLN3nLWBR register should be set to $0000_B$ or $1111_B$.)

In addition, in LIN self-test mode, the following functions are not supported.

- LIN wake-up mode
- Frame separate mode
- Multi-byte response transmission/reception
- LIN slave mode (Auto baud rate)
- Frame/response timeout error

Do not use these functions.

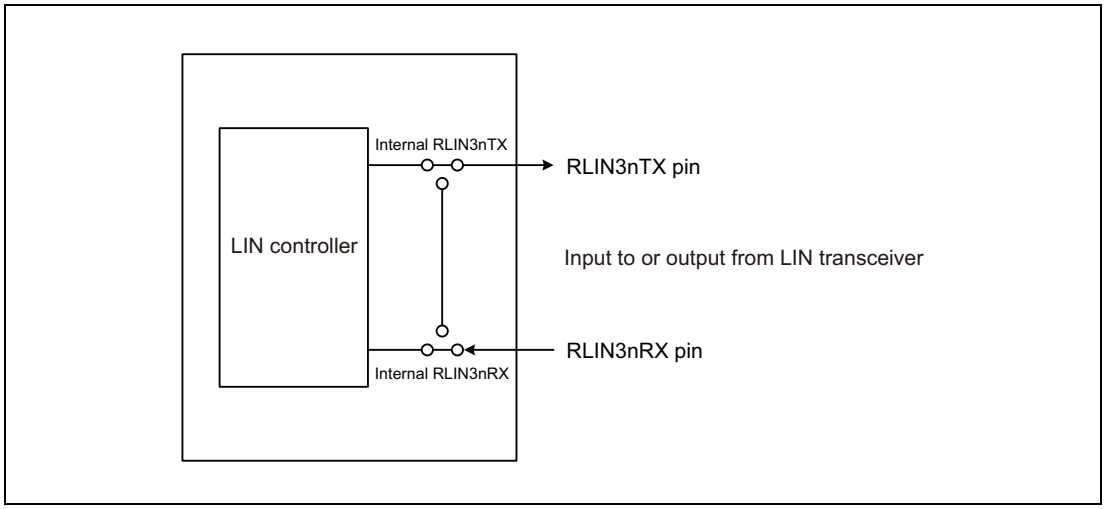



**Figure 19.33 Connection in LIN Reset Mode, LIN Mode, and UART Mode**

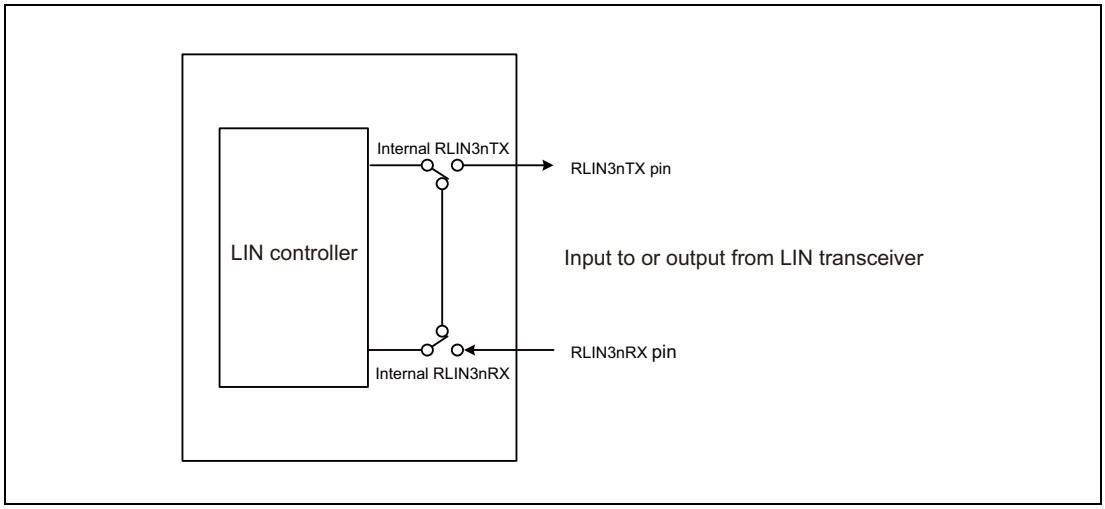



**Figure 19.34 Connection in LIN Self-Test Mode**

### 19.9.1 Transition to LIN Self-Test Mode

Writing to the RLN3nLSTC register makes a transition to the LIN self-test mode.
The LSTM bit in the RLN3nLSTC register set to 1 indicates that the mode has transitioned to the LIN self-test mode.

When transitioning to LIN self-test mode, be sure to execute a specific sequence. In that sequence, information must be consecutively written three times to the LIN self-test control register, as follows:

- Transition to LIN reset mode
  Set the OM0 bit in the RLN3nLCUC register to 0 (LIN reset mode).
  Read the OMM0 bit in the RLN3nLMST register; verify that it is 0 (LIN reset mode).
- Select a LIN mode
  LMD bits in RLN3nLMD = $00_B$ (LIN master mode) or $11_B$ (LIN slave mode [fixed baud rate])
- 1st write: RLN3nLSTC register = $1010\ 0111_B$ ($A7_H$)
- 2nd write: RLN3nLSTC register = $0101\ 1000_B$ ($58_H$)
- 3rd write: RLN3nLSTC register = $0000\ 0001_B$ ($01_H$)
- Verify the transition to LIN self-test mode
  Read the LSTM bit in the RLN3nLSTC register; verify that it is 1 (LIN self-test mode).

If the key of the 1st write ($A7_H$) is written twice by mistake, the transition to LIN self-test mode is aborted. The above sequence should be retried from the 1st write step. In addition, if a write to another LIN-related register is performed during transition to LIN self-test mode (three consecutive write operations to the RLN3nLSTC register), the transition is also aborted.

### 19.9.2 Transmission in LIN Master Self-Test Mode

To execute a self-test on LIN master transmission, perform the following procedure:

- Configure the baud rate, noise filter, and interrupt output related registers.
  RLN3nLWBR register = 0000 $xxxx_B$*1
  RLN3nLBRP0 register = xxxx $xxxx_B$*1
  RLN3nLBRP1 register = xxxx $xxxx_B$*1
  RLN3nLMD register = 00xx $xx00_B$*1
- Configure the interrupt enable and error enable related registers.
  RLN3nLIE register = 0000 $xxxx_B$*2
  RLN3nLEDE register = x000 $x0xx_B$
- Configure the break field and space related registers.
  RLN3nLBFC register = 00xx $xxxx_B$
  RLN3nLSC register = 00xx $0xxx_B$
- Exit the LIN reset mode.
  Write $11_B$ to the OM1 and OM0 bits in the RLN3nLCUC register, and check that the OMM1 and OMM0 bits in the RLN3nLMST register are set to $11_B$.
- Configure the transmission frame related registers.
  RLN3nLDFC register = 00x1 $xxxx_B$
  RLN3nLIDB register = xxxx $xxxx_B$
  RLN3nLDRB1 to RLN3nLDRB8 registers = xxxx $xxxx_B$
- Start header transmission → response transmission
  Set the FTS bit in the RLN3nLTRC register to 1 (frame transmission or wake-up transmission/ reception started).
  The LIN master self-test mode (transmission) is executed, interrupt is generated, and status and error status are also updated. The checksum is automatically calculated by the LIN/UART interface. To abort the LIN master self-test mode (transmission) being executed, write 0 (LIN reset mode) to the OM0 bit in the RLN3nLCUC register for transition to LIN reset mode.
- When the transmission is completed, the reversed value of the looped-back frame data is stored in the RLN3nLIDB, RLN3nLDBRb (b = 1 to 8), and RLN3nLCBR registers (the data is reversed before being stored because the transmitted value should be compared with the looped-back value). Then, the FTS bit in the RLN3nLTRC register is cleared.
- If the transmission fails to complete due to an error, the applicable error flag is set and the FTS bit in the RLN3nLTRC register is cleared.

**Note:** x: Arbitrary value can be specified.

**Note 1.** The settings of the following registers are not reflected to the operation of the LIN self-test mode: the LPRS bit in the RLN3nLWBR register, the RLN3nLBRP0 register, the RLN3nLBRP1 register and the LCKS bit in the RLN3nLMD register. Therefore, configuration of these registers is not necessary.

**Note 2.** If necessary, configure the related registers described in **Section 7, Exception/Interrupts**.

**Note 3.** When the successful header transmission interrupt and the successful frame transmission interrupt are used in the same interrupt processing, if the software processing of the successful header transmission interrupt is not completed before the generation of the successful frame transmission interrupt, the SHIE bit in the RLN3nLIE register should not be set to 1 (successful header transmission interrupt enabled).

The time required from the set of the successful header transmission flag to the set of the successful frame/wake-up transmission flag is calculated by the following formula.

10 × (number of data bytes + 1) [Tbit]

1 Tbit = 1/frequency of LIN communication clock source × 16

### 19.9.3 Reception in LIN Master Self-Test Mode

To execute a self-test on LIN master reception, perform the following procedure:

- Configure the baud rate, noise filter, and interrupt output related registers.
  RLN3nLWBR register = 0000 $xxxx_B$*1
  RLN3nLBRP0 register = xxxx $xxxx_B$*1
  RLN3nLBRP1 register = xxxx $xxxx_B$*1
  RLN3nLMD register = 00xx $xx00_B$*1
- Configure the interrupt enable and error enable related registers.
  RLN3nLIE register = 0000 $xxxx_B$*2
  RLN3nLEDE register = x000 $x0xx_B$
- Configure the break field and space related registers.
  RLN3nLBFC register = 00xx $xxxx_B$
  RLN3nLSC register = 00xx $0xxx_B$*1
- Exit the LIN reset mode.
  Write $11_B$ to the OM1 and OM0 bits in the RLN3nLCUC register, and check that the OMM1 and OMM0 bits in the RLN3nLMST register are set to $11_B$.
- Configure the reception frame related registers.
  RLN3nLDFC register = 00x0 $xxxx_B$
  RLN3nLIDB register = xxxx $xxxx_B$
  RLN3nLDRB1 to RLN3nLDRB8 registers = xxxx $xxxx_B$
  RLN3nCBR register = xxxx $xxxx_B$
  Since the checksum value to be transmitted is not automatically calculated, perform the calculation and specify the calculated value in the RLN3nLCBR register.
- Start header transmission → response reception
  Set the FTS bit in the RLN3nLTRC register to 1 (frame transmission or wake-up transmission/ reception started).
  The LIN master self-test mode (reception) is executed, interrupt is generated, and status and error status are also updated. To abort the LIN master self-test mode (reception) being executed, write 0 (LIN reset mode) to the OM0 bit in the RLN3nLCUC register for transition to LIN reset mode.
- When the reception is completed, the reversed value of the looped-back frame data is stored in the RLN3nLIDB, RLN3nLDBRb (b = 1 to 8), and RLN3nLCBR registers (the data is reversed before being stored because the specified value should be compared with the looped-back value). Then, the FTS bit in the RLN3nLTRC register is cleared.
- If the reception fails to complete due to an error, the applicable error flag is set and the FTS bit in the RLN3nLTRC register is cleared.

**Note:** x: Arbitrary value can be specified.

**Note 1.** The settings of the following registers are not reflected to the operation of the LIN self-test mode: the LPRS bit in the RLN3nLWBR register, the RLN3nLBRP0 register, the RLN3nLBRP1 register, the LCKS bit in the RLN3nLMD register, and the IBS bit in the RLN3nLSC register. Therefore, configuration of these registers is not necessary.

**Note 2.** If necessary, configure the related registers described in **Section 7, Exception/Interrupts**.

**Note 3.** When the successful header transmission interrupt and the successful frame reception interrupt are used in the same interrupt processing, if the software processing of the successful header transmission interrupt is not completed before the generation of the successful frame reception interrupt, the SHIE bit in the RLN3nLIE register should not be set to 1 (successful header transmission interrupt enabled).
The time required from the set of the successful header transmission flag to the set of the successful frame/wake-up reception flag is calculated by the following formula.

10 × (number of data bytes + 1) [Tbit]

1 Tbit = 1/frequency of LIN communication clock source × 16

### 19.9.4 Transmission in LIN Slave Self-Test Mode

To execute a self-test on LIN slave transmission, perform the following procedure:

- Configure the baud rate, noise filter, and interrupt output related registers.
  RLN3nLWBR register = 0000 $xxx0_B$*1
  RLN3nLBRP0 register = xxxx $xxxx_B$*1
  RLN3nLBRP1 register = xxxx $xxxx_B$*1
  RLN3nLMD register = 00x $x0011_B$
- Configure the interrupt enable and error enable related registers.
  RLN3nLIE register = 0000 $xxxx_B$*2
  RLN3nLEDE register = xx0x $x00x_B$
- Configure the break field and space related registers.
  RLN3nLBFC register = 0000 $000x_B$*3
  RLN3nLSC register = 00xx $0001_B$
- Exit the LIN reset mode.
  Write $11_B$ to the OM1 and OM0 bits in the RLN3nLCUC register, and check that the OMM1 and OMM0 bits in the RLN3nLMST register are set to $11_B$.
- Configure the transmission frame related registers.
  RLN3nLDFC register = 00x1 $xxxx_B$
  RLN3nLIDB register = xxxx $xxxx_B$
  RLN3nLDBR1 to RLN3nLDBR8 registers = xxxx $xxxx_B$
- Header reception → response transmission started
  Set the FTS bit in the RLN3nLTRC register to 1 (header reception or wake-up transmission/ reception started).
  (The header reception and the response transmission are executed in this order, without manipulating the RTS bit in the RLN3nLTRC register.)
  The LIN slave self-test mode (transmission) is executed, interrupt is generated, and status and error status are also updated.
  The checksum is automatically calculated by the LIN/UART interface. To abort the LIN master self-test mode (transmission) being executed, write 0 (LIN reset mode) to the OM0 bit in the RLN3nLCUC register for transition to LIN reset mode.
- When the transmission is completed, the reversed value of the looped-back frame data is stored in the RLN3nLIDB, RLN3nLDBRb (b = 1 to 8), and RLN3nLCBR registers (the data is reversed before being stored because the transmitted value should be compared with the looped-back value). Then, the FTS bit in the RLN3nLTRC register is cleared.

- If the transmission fails to complete due to an error, the applicable error flag is set and the FTS bit in the RLN3nLTRC register is cleared.

**Note:** x: Arbitrary value can be specified.

**Note 1.** The settings of the following registers are not reflected to the operation of the LIN self-test mode: the LPRS bit in the RLN3nLWBR register, the RLN3nLBRP0 register, and the RLN3nLBRP1 register. Therefore, configuration of these registers is not necessary.

**Note 2.** If necessary, configure the related registers described in **Section 7, Exception/Interrupts**

**Note 3.** According to the setting of this register, 9.5-Tbit or 10.5-Tbit width break is output from the internal RLIN3nTX.

**Note 4.** When the successful header reception interrupt and the successful response transmission interrupt are used in the same interrupt processing, if the software processing of the successful header reception interrupt is not completed before the generation of the successful response transmission interrupt, the SHIE bit in the RLN3nLIE register should not be set to 1 (successful header reception interrupt enabled).
The time from setting of the successful header reception flag to setting of the successful response/wake-up transmission flag is calculated by using the following formula.

10 × (number of data bytes + 1) [Tbit]

1 Tbit = 1/frequency of LIN communication clock source × 16

### 19.9.5 Reception in LIN Slave Self-Test Mode

To execute a self-test on LIN slave reception, perform the following procedure:

- Configure the baud rate, noise filter, and interrupt output related registers.
  RLN3nLWBR register = 0000 $xxx0_B$*1
  RLN3nLBRP0 register = xxxx $xxxx_B$*1
  RLN3nLBRP1 register = xxxx $xxxx_B$*1
  RLN3nLMD register = 00xx $0011_B$
- Configure the interrupt enable and error enable related registers.
  RLN3nLIE register = 0000 $xxxx_B$*2
  RLN3nLEDE register = xx0x $x00x_B$
- Configure the break field and space related registers.
  RLN3nLBFC register = 0000 $000x_B$*3
  RLN3nLSC register = 00xx $0001_B$*1
- Exit the LIN reset mode.
  Write $11_B$ to the OM1 and OM0 bits in the RLN3nLCUC register, and check that the OMM1 and OMM0 bits in the RLN3nLMST register are set to $11_B$.
- Configure the reception frame related registers.
  RLN3nLDFC register = 00x0 $xxxx_B$
  RLN3nLIDB register = xxxx $xxxx_B$
  RLN3nLDBR1 to RLN3nLDBR8 registers = xxxx $xxxx_B$
  RLN3nCBR register = xxxx $xxxx_B$
  Since the checksum value to be transmitted is not automatically calculated, users must calculate it and set it to the RLN3nLCBR register. A checksum test can be performed by setting an incorrect checksum value here.
- Header reception → response reception started
  Set the FTS bit in the RLN3nLTRC register to 1 (header reception or wake-up transmission/ reception started).
  (Without any setting of the RTS bit in the RLN3nLTRC register, the header reception and the response reception are executed in this order.)
  The LIN slave self-test mode (reception) is executed, interrupt is generated, and status and error status are also updated. To abort the LIN slave self-test mode (reception) being executed, write 0 (LIN reset mode) to the OM0 bit in the RLN3nLCUC register for transition to LIN reset mode.
- When the reception is completed, the reversed value of the looped-back frame data is stored in the RLN3nLIDB, RLN3nLDBRb (b = 1 to 8), and RLN3nLCBR registers (the data is reversed before being stored because the specified value should be compared with the looped-back value). Then, the FTS bit in the RLN3nLTRC register is cleared.
- If the reception fails to complete due to an error, the applicable error flag is set and the FTS bit in the RLN3nLTRC register is cleared.

**Note:** x: Arbitrary value can be specified.

**Note 1.** The settings of the following registers are not reflected to the operation of the LIN self-test mode: the LPRS bit in the RLN3nLWBR register, the RLN3nLBRP0 register, the RLN3nLBRP1 register, and the IBS bit in the RLN3nLSC register.
Therefore, configuration of these registers is not necessary.

**Note 2.** If necessary, configure the related registers described in **Section 7, Exception/Interrupts**.

**Note 3.** According to the setting of this register, 9.5-Tbit or 10.5-Tbit width break is output from the internal RLIN3nTX.

**Note 4.** When the successful header reception interrupt and the successful response reception interrupt are used in the same interrupt processing, if the software processing of the successful header reception interrupt is not completed before the generation of the successful response reception interrupt, the SHIE bit in the RLN3nLIE register should not be set to 1 (successful header reception interrupt enabled).
The time required from the set of the successful header reception flag to the set of the successful response/wake-up reception flag is calculated by the following formula.

10 × (number of data bytes + 1) [Tbit]

1 Tbit = 1/frequency of LIN communication clock source × 16

### 19.9.6 Exiting LIN Self-Test Mode

To exit LIN self-test mode, perform the following procedure:

- Write 0 (LIN reset mode) to the OM0 bit in the RLN3nLCUC register.
  If the OMM1 and OMM0 bits in the RLN3nLMST register are not $11_B$, write $11_B$ to the OM1 and OM0 bits in the RLN3nLCUC register. After confirming that the OMM1 and OMM0 bits in the RLN3nLMST register are set to $11_B$, transition to LIN reset mode.
- Verify that LIN/UART interface has exited LIN self-test mode.
  Read the LSTM bit in the RLN3nLSTC register; verify that it is 0 (not in LIN self-test mode)
- Verify the transition to LIN reset mode.
  Read the OMM0 bit in the RLN3nLMST register; verify that it is 0 (LIN reset mode).

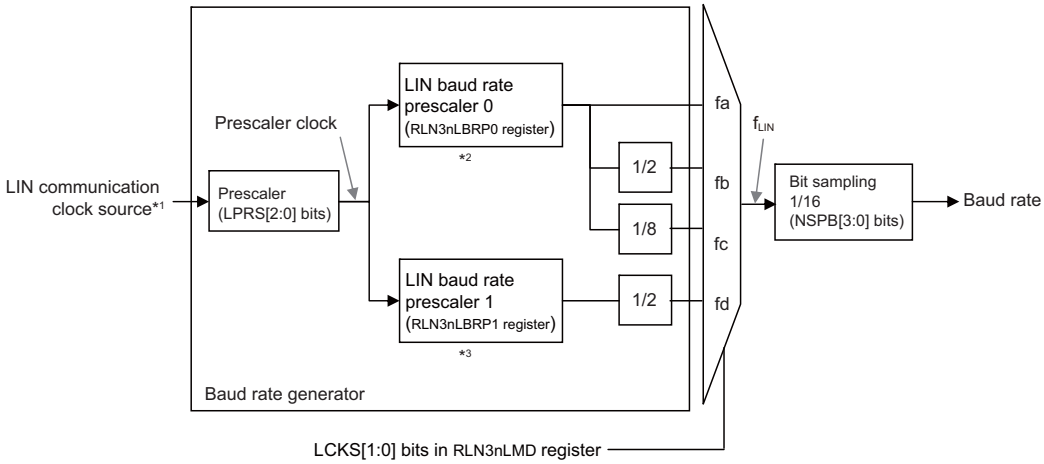
## 19.10 Baud Rate Generator

The prescaler clock is obtained by dividing the LIN communication clock source frequency by the prescaler, and the LIN system clock ($f_{LIN}$) is obtained by dividing the prescaler clock frequency by the baud rate generator. The clock obtained by dividing the LIN system clock frequency ($f_{LIN}$) by the number of samples is the baud rate. The inverse of this baud rate is called the bit time (Tbit).

The LIN/UART interface has two types of baud rate generators. The baud rate generators are switched according to the mode used.

### 19.10.1 LIN Master Mode

**Figure 19.35** shows a block diagram of baud rate generation in LIN master mode.




Note 1. For the LIN communication clock source, see **Section 12.5.3, Clock Domain Settings**.
Note 2. When the value in RLN3nLBRP0 register is N (N = 0 to 255), the clock frequency is divided by N + 1.
Note 3. When the value in RLN3nLBRP1 register is M (M = 0 to 255), the clock frequency is divided by M + 1.

**Figure 19.35 Block Diagram of Baud Rate Generation in LIN Master Mode**

By setting the RLN3nLBRP0 register so that fa is 307200 Hz (= 19200 × 16), the resulting bit rates are fa = 19200 × 16, fb = 9600 × 16, and fc = 2400 × 16. These bit rates are frequency-divided by 16 in the bit timing generator, enabling baud rates of 19200 bps, 9600 bps and 2400 bps, to be generated. Also, by configuring the RLN3nLBRP1 register so that fd is 166672 Hz (= 10417 × 16), the resulting bit rate is fd = 10417 × 16. This bit rate is frequency-divided by 16 in the bit timing generator, enabling 10417 bps to be generated.

Baud rate of LIN master

= {Frequency of LIN communication clock source} × (RLN3nLWBR.LPRS[2:0] selected clock)
÷ (RLN3nLBRP0+ 1) ÷ 16 [bps] (When fa is selected for $f_{LIN}$)

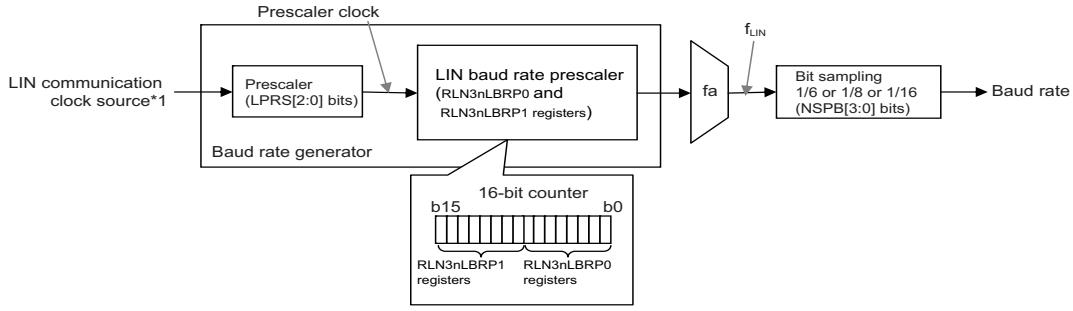
= {Frequency of LIN communication clock source} × (RLN3nLWBR.LPRS[2:0] selected clock)
÷ (RLN3nLBRP0 + 1) ÷ 2 ÷ 16 [bps] (When fb is selected for $f_{LIN}$)

= {Frequency of LIN communication clock source} × (RLN3nLWBR.LPRS[2:0] selected clock)
÷ (RLN3nLBRP0 + 1) ÷ 8 ÷ 16 [bps] (When fc is selected for $f_{LIN}$)

= {Frequency of LIN communication clock source} × (RLN3nLWBR.LPRS[2:0] selected clock)
÷ (RLN3nLBRP1 + 1) ÷ 2 ÷ 16 [bps] (When fd is selected for $f_{LIN}$)

### 19.10.2 LIN Slave Mode

**Figure 19.36** shows a block diagram of baud rate generation in LIN slave mode.




Note 1. For the LIN communication clock source, see **Section 12.5.3, Clock Domain Settings**.

**Figure 19.36 Block Diagram of Baud Rate Generation in LIN Slave Mode**

In LIN slave mode (auto baud rate), the baud rate can be specified in the range of 1 kbps to 20 kbps. Configure the prescaler clock as follows according to the target baud rate:

| [Target baud rate] | [Prescaler clock] |
|---|---|
| 1 kbps to 20 kbps | : 4MHz *1 |
| 1 kbps to 2.4 kbps (excluding 2.4 kbps) | : 4MHz |
| 2.4 kbps to 20 kbps | : 8 MHz to 12 MHz |

**Note 1.** Use the clock with NSPB[3:0] bits in the RLN3nLWBR register set to "$0011_B$" (4 samplings).

The formula for baud rate is described below.

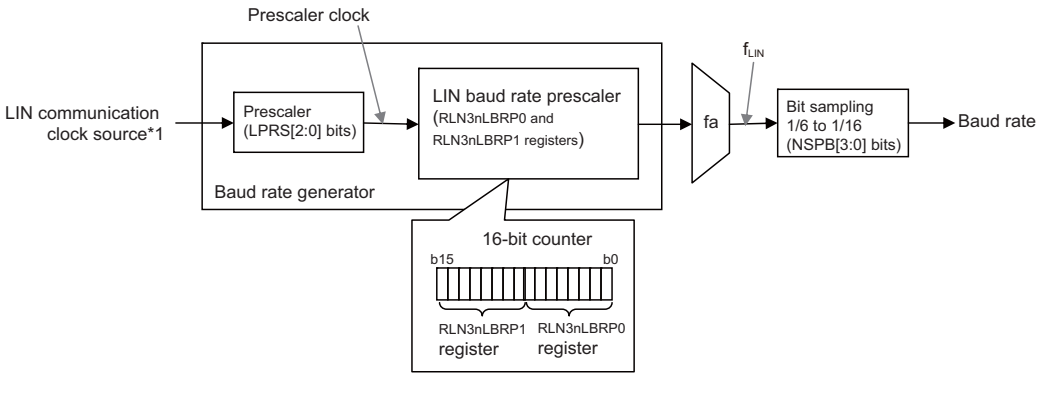
Baud rate of LIN slave
= {Frequency of LIN communication clock source} × (RLN3nLWBR.LPRS[2:0] selected clock)
÷ (RLN3nLBRP01 + 1) ÷ 16 [bps] ([Fixed baud rate])
= {Frequency of LIN communication clock source} × (RLN3nLWBR.LPRS[2:0] selected clock)
÷ (RLN3nLBRP01 + 1) ÷ 4 or 8 [bps] ([Auto baud rate])

**NOTE**

For a LIN slave with fixed baud rate, set the NSPB[3:0] bit to "$0000_B$" (16 samplings) or "$1111_B$" (16 samplings). For a LIN slave with auto baud rate, set the NSPB[3:0] bits to "$0011_B$" (4 samplings) or "$0100_B$" (8 samplings).

### 19.10.3 UART Mode

**Figure 19.37** shows a block diagram of baud rate generation in UART mode.




Note 1. For the LIN communication clock source, see **Section 12.5.3, Clock Domain Settings**.

**Figure 19.37 Block Diagram of Baud Rate Generation in UART Mode**

UART baud rate is calculated with the following formula:

UART baud rate
= {LIN communication clock source frequency} × (RLN3nLWBR.LPRS[2:0] selected clock) ÷ (RLN3nLBRP01 + 1) ÷ {RLN3nLWBR.NSPB[3:0] selected count} [bps]

## 19.11 Noise Filter

The LIN/UART interface has a noise filter for reducing erroneous receiving of data due to noise. By setting the LRDNFS bit in the RLN3nLMD register to 0 (use the noise filter), the noise filter is activated. The noise filter samples the level of the synchronized RLIN3nRX with the prescaler clock, and outputs the sampling value determined by a 3-sampling majority rule. The value of each bit of the receive data is determined based on the noise filter output.

**Figure 19.38** shows the configuration of the noise filter, **Figure 19.39** shows an example of a noise filter circuit, and **Figure 19.40** shows the determination of the received data when the noise filter is used.

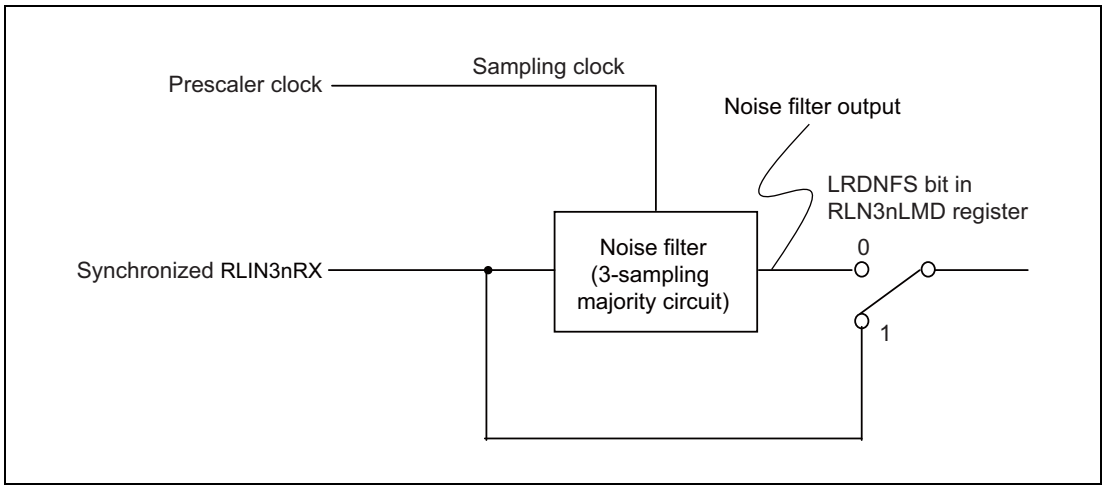



**Figure 19.38 Configuration of Noise Filter**

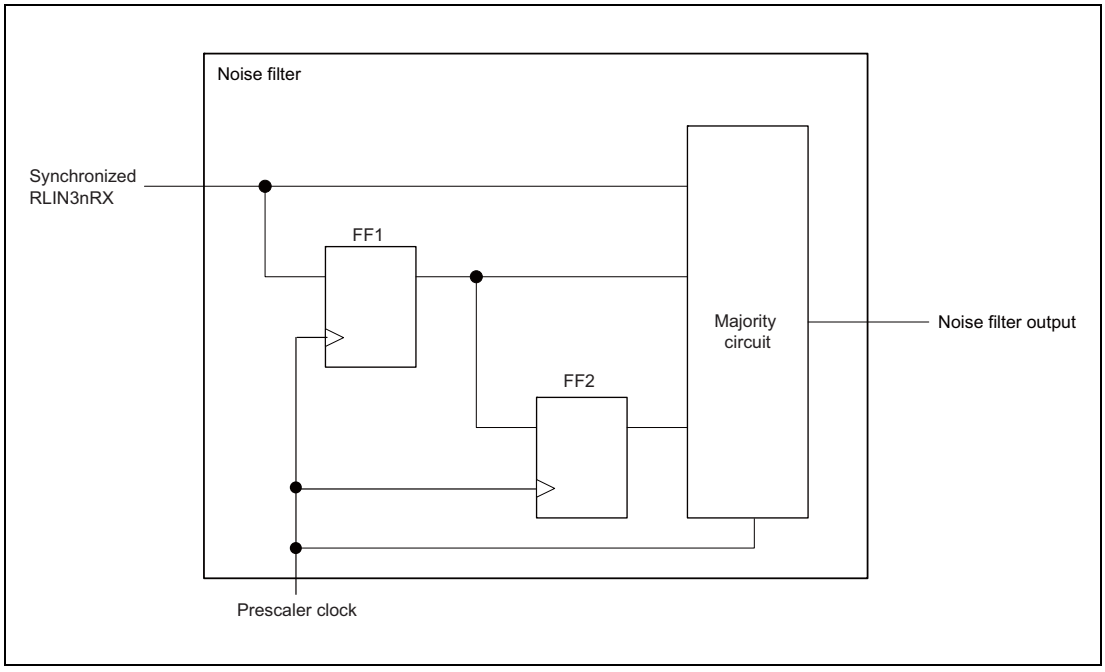



**Figure 19.39 Example of Noise Filter Circuit**

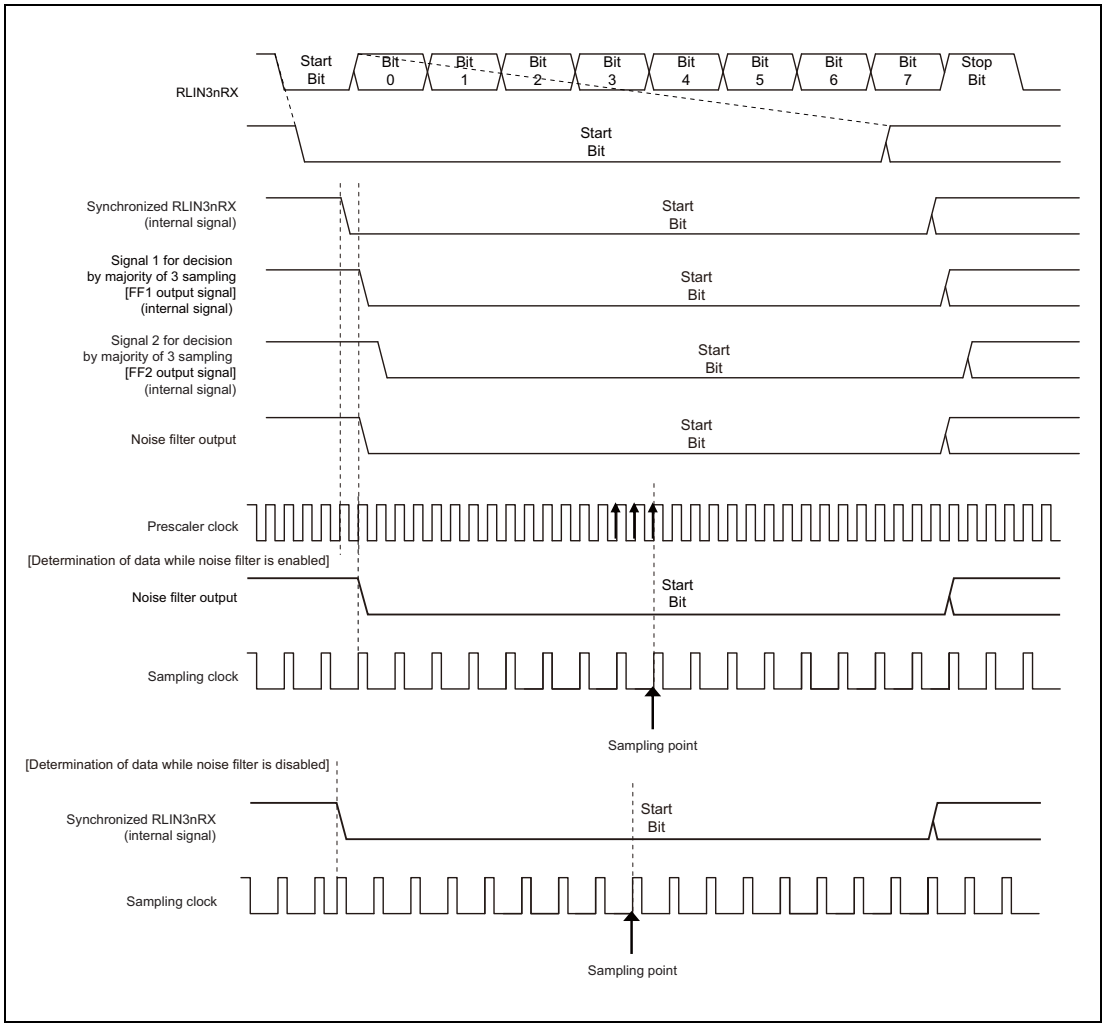



**Figure 19.40 Determination of Received Data when Noise Filter is Used**

# Section 20 I2C Bus Interface (RIIC)

This section contains a generic description of the I2C Bus Interface (RIIC).

The first part of this section describes all RH850/F1K specific properties, such as the number of units, register base addresses, etc. The remainder of the section describes the functions and registers of RIIC.

## 20.1 Features of RH850/F1K RIIC

### 20.1.1 Number of Units and Channels

This microcontroller has the following number of RIIC units.

Each RIIC unit has one channel interface. “Number of channels” is used with the same meaning as “number of units” in this section.

**Table 20.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of Units | 1 | | |
| Name | RIICn (n = 0) | | |

**Table 20.2 Index**

| Index | Description |
|---|---|
| n | Throughout this section, the individual RIIC units are identified by the index “n” (n = 0): for example, RIICnCR1 is the I2C bus control register1. |

### 20.1.2 Register Base Address

RIIC base addresses are listed in the following table.

RIIC register addresses are given as offsets from the base addresses.

**Table 20.3 Register Base Address**

| Base Address Name | Base Address |
|---|---|
| <RIIC0_base> | FFCA 0000H |

### 20.1.3 Clock Supply

The RIIC clock supply is shown in the following table.

**Table 20.4 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| RIICn | PCLK*1 | CKSCLK_IIIC | Communication clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CKSCLK_IIIC | |

Note 1. Set the period of PCLK no greater than 1/2 of the width at high level of the SCL clock.

### 20.1.4 Interrupt Requests

RIIC interrupt requests are listed in the following table.

**Table 20.5 Interrupt Requests**

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **RIIC0** | | | |
| INTRIICnEE | RIIC communication error / event generation interrupt | 77 | — |
| INTRIICnRI | RIIC receive end interrupt | 78 | 20 |
| INTRIICnTI | RIIC transmit data empty interrupt | 76 | 19 |
| INTRIICnTEI | RIIC transmit end interrupt | 79 | — |

### 20.1.5 Reset Sources

RIIC reset sources are listed in the following table. RIIC is initialized by these reset sources.

**Table 20.6 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| RIIC0 | All reset sources (ISORES) |

### 20.1.6 External Input/Output Signals

External input/output signals of RIIC are listed below.

**Table 20.7 External Input/Output Signals**

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **RIIC0** | | |
| RIICnSCL | Serial clock I/O pin | RIIC0SCL |
| RIICnSDA | Serial data I/O pin | RIIC0SDA |

When using these ports, the PBDCn register for the corresponding port and the corresponding bit in the PODCn register must be set to 1.

# 20.2 Overview

## 20.2.1 Functional Overview

### Communications format

- I2C bus format
- Master mode or slave mode selectable
- Automatic securing of the various set-up times, hold times, and bus-free times according to the specified transfer rate

### Transfer rate

Up to 400 kbps

### SCL clock

- For master operation, the duty cycle of the SCL clock is selectable in the following range:
  - 0% < Duty < 100%

### Issuing and detecting conditions

Start, restart, and stop conditions are automatically generated. Start conditions (including restart conditions) and stop conditions are detectable.

### Slave address

- Up to three slave-address settings can be made.
- Seven- and ten-bit address formats are supported (along with the use of both at once).
- General call addresses and device ID addresses are detectable.

### Acknowledgement

- For transmission, the acknowledge bit is automatically loaded
  - Transfer of the next data for transmission can be automatically suspended on detection of a not-acknowledge bit.
- For reception, the acknowledge bit is automatically transmitted
  - If a wait between the eighth and ninth clock cycles has been selected, software control of the value in the acknowledge field in response to the received value is possible.

### Wait function

- In reception, the following periods of waiting can be obtained by holding the clock signal (SCL) at the low level:
  - Waiting between the eighth and ninth clock cycles
  - Waiting between the ninth clock cycle and the first clock cycle of the next transfer (WAIT function)

### SDA output delay function

Timing of the output of transmitted data, including the acknowledge bit, can be delayed.

### Arbitration

- For multi-master operation
  - Operation to synchronize the SCL (clock) signal in cases of conflict with the SCL signal from another master is possible.
  - When issuing the start condition would create conflict on the bus, loss of arbitration is detected by testing for non-matching between the internal signal for the SDA line and the level on the SDA line.
  - In master operation, loss of arbitration is detected by testing for non-matching of internal and line levels for transmit data.
- Loss of arbitration due to detection of the start condition while the bus is busy can be detected (to prevent the issuing of double start conditions).
- Loss of arbitration in transfer of a not-acknowledge bit due to the internal signal for the SDA line and the level on the SDA line not matching can be detected.
- Loss of arbitration due to non-matching of internal and line levels for data can be detected in slave transmission.

### Timeout function

The internal time-out function is capable of detecting long-interval stop of the SCL (clock signal).

### Noise removal

The interface incorporates digital noise filters for both the SCL and SDA signals, and the width for noise cancellation by the filters is adjustable by software.

### Interrupt sources

- Four sources:
  - Error in transfer or occurrence of events (detection of arbitration loss, NACK, time-out, a start condition including a restart condition, or a stop condition)
  - Reception complete (including matching with a slave address)
  - Transmit-data-empty (including matching with a slave address)
  - Transmission complete

## 20.2.2 Block Diagram

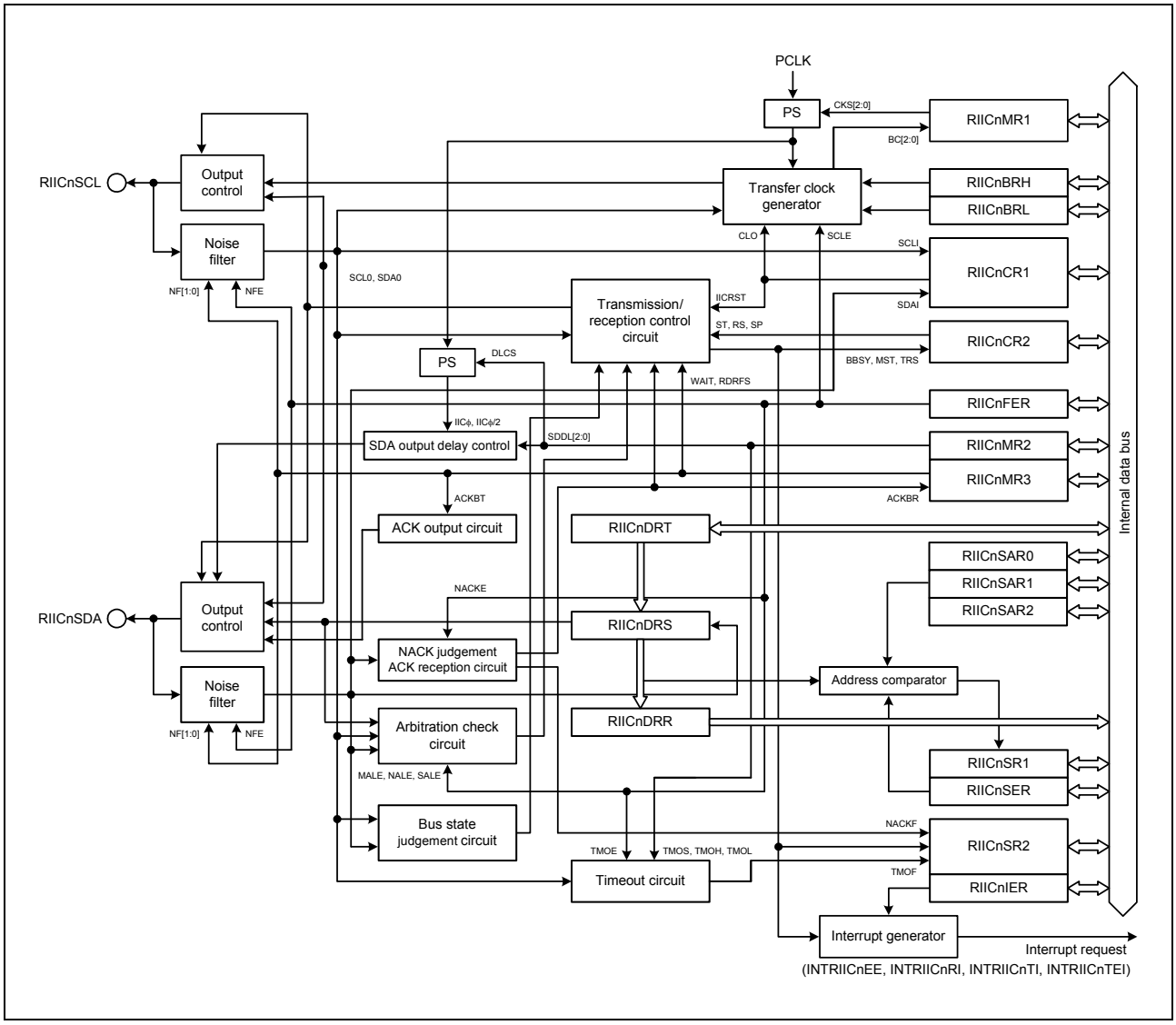



**Figure 20.1 Block Diagram of RIIC**

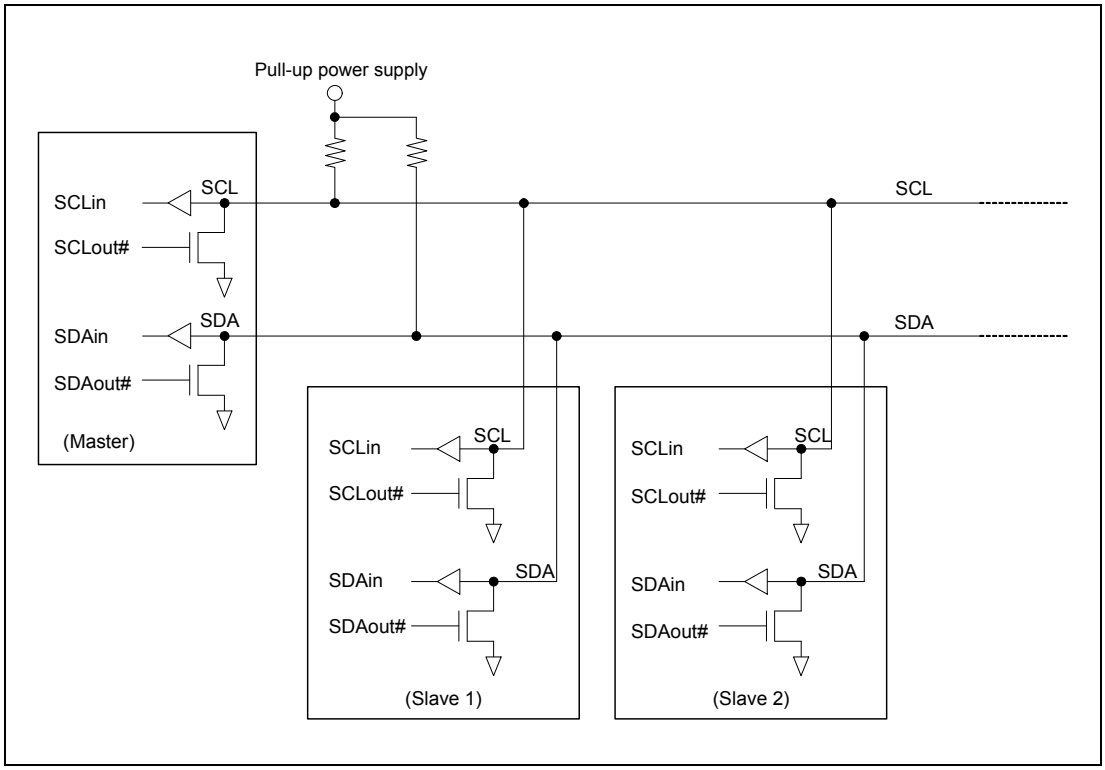



**Figure 20.2 Connections to the External Circuit by the I/O Pins (I2C Bus Configuration Example)**

# 20.3 Registers

## 20.3.1 List of Registers

RIIC registers are listed in the table below.

For details about <RIICn_base>, see **Section 20.1.2, Register Base Address**.

**Table 20.8 List of Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| RIICn | I2C Bus Control Register 1 | RIICnCR1 | <RIICn_base> + $0000_H$ |
| | I2C Bus Control Register 2 | RIICnCR2 | <RIICn_base> + $0004_H$ |
| | I2C Bus Mode Register 1 | RIICnMR1 | <RIICn_base> + $0008_H$ |
| | I2C Bus Mode Register 2 | RIICnMR2 | <RIICn_base> + $000C_H$ |
| | I2C Bus Mode Register 3 | RIICnMR3 | <RIICn_base> + $0010_H$ |
| | I2C Bus Function Enable Register | RIICnFER | <RIICn_base> + $0014_H$ |
| | I2C Bus Status Enable Register | RIICnSER | <RIICn_base> + $0018_H$ |
| | I2C Bus Interrupt Enable Register | RIICnIER | <RIICn_base> + $001C_H$ |
| | I2C Bus Status Register 1 | RIICnSR1 | <RIICn_base> + $0020_H$ |
| | I2C Bus Status Register 2 | RIICnSR2 | <RIICn_base> + $0024_H$ |
| | I2C Slave Address Register 0 | RIICnSAR0 | <RIICn_base> + $0028_H$ |
| | I2C Slave Address Register 1 | RIICnSAR1 | <RIICn_base> + $002C_H$ |
| | I2C Slave Address Register 2 | RIICnSAR2 | <RIICn_base> + $0030_H$ |
| | I2C Bus Bit Rate Low-Level Register | RIICnBRL | <RIICn_base> + $0034_H$ |
| | I2C Bus Bit Rate High-Level Register | RIICnBRH | <RIICn_base> + $0038_H$ |
| | I2C Bus Transmit Data Register | RIICnDRT | <RIICn_base> + $003C_H$ |
| | I2C Bus Receive Data Register | RIICnDRR | <RIICn_base> + $0040_H$ |
| | I2C Bus Shift Register | RIICnDRS | — |

## 20.3.2 RIICnCR1 — I2C Bus Control Register 1

**Access:** RIICnCR1 register can be read or written in 32-bit units.
RIICnCR1L register can be read or written in 16-bit units.
RIICnCR1LL register can be read or written in 8-bit units.

**Address:** RIICnCR1: <RIICn_base> + $0000_H$

RIICnCR1L: <RIICn_base> + $0000_H$

RIICnCR1LL: <RIICn_base> + $0000_H$

**Value after reset:** 0000 $001F_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ICE | IICRST | CLO | SOWP | SCLO | SDAO | SCLI | SDAI |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | W | R/W | R/W | R | R |

**Table 20.9 RIICnCR1 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 7 | ICE | I2C Bus Interface Enable<br>0: Disabled (the RIICnSCL and RIICnSDA pins are not driven).<br>1: Enabled (the RIICnSCL and RIICnSDA pins are driven).<br>(This bit selects an RIIC reset or internal reset in combination with the IICRST bit.) |
| 6 | IICRST | I2C Bus Interface Internal Reset<br>0: Clears the RIIC reset or internal reset.<br>1: Initiates the RIIC reset or internal reset.<br>(Clears the bit counter and the SCL/SDA output latch) |
| 5 | CLO | Extra SCL Clock Cycle Output<br>0: Does not output an extra SCL clock cycle (default).<br>1: Outputs an extra SCL clock cycle.<br>(The CLO bit is cleared automatically after one clock cycle is output.) |
| 4 | SOWP | SCLO/SDAO Write Protect<br>0: Bits SCLO and SDAO can be written.<br>1: Bits SCLO and SDAO are protected.<br>(This bit is read as 1.) |
| 3 | SCLO | SCL Output Control/Monitor<br>• Read:<br>0: The RIIC has driven the RIICnSCL pin low.<br>1: The RIIC has released the RIICnSCL pin.<br>• Write:<br>0: The RIIC drives the RIICnSCL pin low.<br>1: The RIIC releases the RIICnSCL pin. |

Table 20.9 RIICnCR1 Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 2 | SDAO | SDA Output Control/Monitor<br>• Read:<br>0: The RIIC has driven the RIICnSDA pin low.<br>1: The RIIC has released the RIICnSDA pin.<br>• Write:<br>0: The RIIC drives the RIICnSDA pin low.<br>1: The RIIC releases the RIICnSDA pin. |
| 1 | SCLI | SCL Line Monitor<br>0: RIICnSCL line is low.<br>1: RIICnSCL line is high. |
| 0 | SDAI | SDA Line Monitor<br>0: RIICnSDA line is low.<br>1: RIICnSDA line is high. |

### SDAO Bit (SDA Output Control/Monitor) and SCLO Bit (SCL Output Control/Monitor)

These bits are used to directly control the RIICnSDA and RIICnSCL signals output from the RIIC.

When writing to these bits, also write 0 to the SOWP bit.

The result of setting these bits is input to the RIIC via the input buffer. When slave mode is selected, a START condition may be detected and the bus may be released depending on the bit settings.

Do not rewrite these bits during a START condition, STOP condition, repeated START condition, or during transmission or reception. Operation after rewriting under the above conditions is not guaranteed.

When reading these bits, the state of signals output from the RIIC can be read.

### CLO Bit (Extra SCL Clock Cycle Output)

This bit is used to output an extra SCL clock cycle for debugging or error processing.

Normally, set the bit to 0. Setting the bit to 1 in a normal communication state causes a communication error.

For details on this function, see **Section 20.13.2, Extra SCL Clock Cycle Output Function**.

### IICRST Bit (I2C Bus Interface Internal Reset)

This bit is used to reset the internal states of the RIIC.

Setting this bit to 1 initiates an RIIC reset or internal reset.

Whether an RIIC reset or internal reset is initiated is determined according to the combination with the ICE bit. **Table 20.10** lists the types of RIIC reset.

The RIIC reset resets all registers (except ICE and IICRST) including the RIICnCR2.BBSY flag and internal states of the RIIC, and the internal reset resets the bit counter (RIICnMR1.BC[2:0] bits), the I2C bus shift register (RIICnDRS), and the I2C bus status registers (RIICnSR1 and RIICnSR2) as well as the internal states of the RIIC. For the reset conditions for each register, see **Section 20.14, Reset Function of RIIC**.

An internal reset initiated with the IICRST bit set to 1 during operation (with the ICE bit set to 1) resets the internal states of the RIIC without initializing the port settings and the control and setting registers of the RIIC when the bus or RIIC hangs up due to a communication errorr, etc.

If the RIIC hangs up in a low level output state, resetting the internal states cancels the low level output state and releases the bus with the RIICnSCL pin and RIICnSDA pin at a high impedance.

**CAUTION**

If an internal reset is initiated using the IICRST bit for a bus hang-up occurred during communication with the master device in slave mode, the states may become different between the slave device and the master device (due to the difference in the bit counter information). For this reason, do not initiate an internal reset in slave mode, but initiate restoration processing from the master device. If an internal reset is necessary because the RIIC hangs up with the SCL line in a low level output state in slave mode, initiate an internal reset and then issue a restart condition from the master device or resume communication from the start condition issuance after issuing a stop condition. If communication is restarted by initiating a reset solely in the slave device without issuing a start condition or restart condition from the master device, synchronization will be lost because the master and slave devices operate asynchronously.

**Table 20.10 RIIC Resets**

| IICRST | ICE | State | Specifications |
|---|---|---|---|
| 1 | 0 | RIIC reset | Resets all registers (except ICE and IICRST) and internal states of the RIIC. |
| | 1 | Internal reset | Reset the RIICnMR1.BC[2:0] bits, and the RIICnSR1, RIICnSR2, RIICnDRS registers and the internal states of the RIIC. |

### ICE Bit (I2C Bus Interface Enable)

The ICE bit selects driving or non-driving of the RIICnSCL and RIICnSDA pins. Moreover, this bit can perform two types of reset in combination with the IICRST bit. For the types of reset, see **Table 20.10, RIIC Resets**.

Set the ICE bit to 1 when using RIIC. Setting the ICE bit to 1 selects driving of the RIICnSCL and RIICnSDA pins.

Set the ICE bit to 0 when RIIC is not to be used. Clearing the ICE bit to 0 stops driving of the RIICnSCL and RIICnSDA pins.

**CAUTION**

Though the output from RIICnSDA or RIICnSCL is disabled while the ICE bit is 0, the input to RIICnSDA or RIICnSCL is enabled. The RIICnSCL and RIICnSDA pin functions should not be assigned to the RIIC. If assigned, it causes the slave addresses to be compared.

## 20.3.3 RIICnCR2 — I2C Bus Control Register 2

**Access:** RIICnCR2 register can be read or written in 32-bit units.
RIICnCR2L register can be read or written in 16-bit units.
RIICnCR2LL register can be read or written in 8-bit units.

**Address:** RIICnCR2: <RIICn_base> + $0004_H$

RIICnCR2L: <RIICn_base> + $0004_H$

RIICnCR2LL: <RIICn_base> + $0004_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | BBSY | MST | TRS | — | SP | RS | ST | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W | R/W | R |

**Table 20.11 RIICnCR2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 7 | BBSY | Bus Busy Detection Flag<br>0: The I2C bus is released (the bus is free).<br>1: The I2C bus is occupied (the bus is busy). |
| 6 | MST*1 | Master/Slave Mode<br>0: Slave mode<br>1: Master mode |
| 5 | TRS*1 | Transmit/Receive Mode<br>0: Receive mode<br>1: Transmit mode |
| 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | SP | Stop Condition Issuance Request<br>0: Does not request to issue a stop condition.<br>1: Requests to issue a stop condition. |
| 2 | RS | Restart Condition Issuance Request<br>0: Does not request to issue a restart condition.<br>1: Requests to issue a restart condition. |
| 1 | ST | Start Condition Issuance Request<br>0: Does not request to issue a start condition.<br>1: Requests to issue a start condition. |
| 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Note 1. When the RIICnMR1.MTWP bit is set to 1, the MST and TRS bits can be written to.

### ST Bit (Start Condition Issuance Request)

This bit is used to request transition to master mode and issuance of a start condition.

When this bit is set to 1 to request to issue a start condition, a start condition is issued when the BBSY flag is set to 0 (bus free).

For details on the start condition issuance, see **Section 20.12, Start Condition/Restart Condition/Stop Condition Issuing Function**.

[Setting condition]

When 1 is written to the ST bit

[Clearing conditions]

- When 0 is written to the ST bit
- When a start condition has been issued
- When the RIICnSR2.AL (arbitration-lost) flag is set to 1
- When 1 is written to the RIICnCR1.IICRST bit to apply an RIIC reset or an internal reset

**CAUTION**

Set the ST bit to 1 (start condition issuance request) when the BBSY flag is set to 0 (bus free).
Note that arbitration may be lost as the start condition issuance error if the ST bit is set to 1 (start condition issuance request) when the BBSY flag is set to 1 (bus busy).

### RS Bit (Restart Condition Issuance Request)

This bit is used to request that a restart condition be issued in master mode.

When this bit is set to 1 to request to issue a restart condition, a restart condition is issued when the BBSY flag is set to 1 (bus busy) and the MST bit is set to 1 (master mode).

For details on the restart condition issuance, see **Section 20.12, Start Condition/Restart Condition/Stop Condition Issuing Function**.

[Setting condition]

When 1 is written to the RS bit with the RIICnCR2.BBSY flag set to 1

[Clearing conditions]

- When 0 is written to the RS bit
- When a restart condition has been issued or a start condition is detected
- When a stop condition is detected
- When the RIICnCR2.AL (arbitration-lost) flag is set to 1
- When 1 is written to the RIICnCR1.IICRST bit to apply an RIIC reset or an internal reset

**CAUTIONS**

1. Do not set the RS bit to 1 while issuing a stop condition.
2. It is commended to issue a restart condition in master transmit mode. If the RS bit is set to 1 (restart condition issuance request) in mode other than master mode, the restart condition is not issued in this mode but the RS bit remains set. If the operating mode changes to master mode with the bit not being cleared, the restart condition may be issued.

### SP Bit (Stop Condition Issuance Request)

This bit is used to request that a stop condition be issued in master mode.

When this bit is set to 1 to request to issue a stop condition, a stop condition is issued when the BBSY flag is set to 1 (bus busy) and the MST bit is set to 1 (master mode).

For details on the stop condition issuance, see **Section 20.12, Start Condition/Restart Condition/Stop Condition Issuing Function**.

[Setting condition]

When 1 is written to the SP bit with both the RIICnCR2.BBSY flag and the RIICnCR2.MST bit set to 1

[Clearing conditions]

- When 0 is written to the SP bit
- When a stop condition has been issued or a stop condition is detected
- When the RIICnSR2.AL (arbitration-lost) flag is set to 1
- When a start condition and a restart condition are detected
- When 1 is written to the RIICnCR1.IICRST bit to apply an RIIC reset or an internal reset

**CAUTIONS**

1. Writing to the SP bit is not possible while the setting of the BBSY flag is 0 (bus free).
2. Do not set the SP bit to 1 while a restart condition is being issued.

### TRS Bit (Transmit/Receive Mode)

This bit indicates transmit or receive mode.

The RIIC is in receive mode when the TRS bit is set to 0 and is in transmit mode when the bit is set to 1. Combination of this bit and the MST bit indicates the operating mode of the RIIC.

The value of the TRS bit is automatically changed to the value for transmission mode or reception mode (1 or 0) by detection or issuing of a start condition, setting or clearing of the R/W# bit, etc. Although writing to the TRS bit is possible when the RIICnMR1.MTWP bit is set to 1, writing to this bit is not necessary during normal usage.

[Setting conditions]

- When a start condition is issued normally according to the start condition issuance request (when a start condition is detected with the ST bit set to 1)
- When the address received in slave mode matches the address enabled in RIICnSER, with the R/W# bit set to 1
- When 1 is written to the TRS bit with the RIICnMR1.MTWP bit set to 1

[Clearing conditions]

- When a stop condition is detected
- The RIICnSR2.AL (arbitration-lost) flag being set to 1
- In master mode, reception of a slave address to which an R/W# bit with the value 1 is appended
- In slave transmit mode, a restart condition is detected (a restart condition is detected with RIICnCR2.BBSY = 1 and RIICnCR2.MST = 0)

- When 0 is written to the TRS bit with the RIICnMR1.MTWP bit set to 1
- When 1 is written to the RIICnCR1.IICRST bit to apply an RIIC reset or an internal reset

### MST Bit (Master/Slave Mode)

This bit indicates master or slave mode.

The RIIC is in slave mode when the MST bit is set to 0 and is in master mode when the bit is set to 1. Combination of this bit and the TRS bit indicates the operating mode of the RIIC.

The value of the MST bit is automatically changed to the value for master mode or slave mode (1 or 0) by detection or issuing of a start condition, etc. Although writing to the MST bit is possible when the RIICnMR1.MTWP bit is set to 1, writing to this bit is not necessary during normal usage.

[Setting conditions]

- When a start condition is issued normally according to the start condition issuance request (when a start condition is detected with the ST bit set to 1)
- When 1 is written to the MST bit with the RIICnMR1.MTWP bit set to 1

[Clearing conditions]

- When a stop condition is detected
- When the RIICnSR2.AL (arbitration-lost) flag is set to 1
- When 0 is written to the MST bit with the RIICnMR1.MTWP bit set to 1
- When 1 is written to the RIICnCR1.IICRST bit to apply an RIIC reset or an internal reset

### BBSY Flag (Bus Busy Detection)

The BBSY flag indicates whether the I2C bus is occupied (bus busy) or released (bus free).

This bit is set to 1 when the SDA line changes from high to low under the condition of SCL = high, assuming that a start condition has been issued.

When the SDA line changes from low to high under the condition of SCL = high, this bit is cleared to 0 after the bus free time (specified in RIICnBRL) start condition is not detected, assuming that a stop condition has been issued.

[Setting condition]

When a start condition is detected

[Clearing conditions]

- When the bus free time (specified in RIICnBRL) start condition is not detected after detecting a stop condition
- When 1 is written to the RIICnCR1.IICRST bit with the RIICnCR1.ICE bit set to 0 (RIIC reset)

**CAUTION**

- When an internal reset is applied while the bus is free after detection of a stop condition, the setting of the BBSY flag is 0 while the bus is free following de-assertion of the internal reset signal.
- When an internal reset is applied while the bus is not free, the BBSY flag is not cleared.

## 20.3.4 RIICnMR1 — I2C Bus Mode Register 1

**Access:** RIICnMR1 register can be read or written in 32-bit units.
RIICnMR1L register can be read or written in 16-bit units.
RIICnMR1LL register can be read or written in 8-bit units.

**Address:** RIICnMR1: <RIICn_base> + $0008_H$

RIICnMR1L: <RIICn_base> + $0008_H$

RIICnMR1LL: <RIICn_base> + $0008_H$

**Value after reset:** $0000\ 0008_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | MTWP | CKS[2:0] | | | BCWP | BC[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | W | R/W | R/W | R/W |

**Table 20.12 RIICnMR1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 7 | MTWP | MST/TRS Write Protect<br>0: Disables writing to the RIICnCR2.MST and TRS bits.<br>1: Enables writing to the RIICnCR2.MST and TRS bits. |
| 6 to 4 | CKS[2:0] | Internal Reference Clock Selection (IICϕ)<br>b6 b4<br>0 0 0: PCLK/1 clock<br>0 0 1: PCLK/2 clock<br>0 1 0: PCLK/4 clock<br>0 1 1: PCLK/8 clock<br>1 0 0: PCLK/16 clock<br>1 0 1: PCLK/32 clock<br>1 1 0: PCLK/64 clock<br>1 1 1: PCLK/128 clock |
| 3 | BCWP*1 | BC Write Protect<br>0: Enables a value to be written in the BC[2:0] bits. (This bit is read as 1.)<br>1: Protects the BC[2:0] bits. |
| 2 to 0 | BC[2:0] | Bit Counter<br>b2 b0<br>0 0 0: 9 bits<br>0 0 1: 2 bits<br>0 1 0: 3 bits<br>0 1 1: 4 bits<br>1 0 0: 5 bits<br>1 0 1: 6 bits<br>1 1 0: 7 bits<br>1 1 1: 8 bits |

Note 1. When rewriting the BC[2:0] bits, write 0 to the BCWP bit simultaneously.

### BC[2:0] Bits (Bit Counter)

These bits function as a counter that indicates the number of bits remaining to be transferred at the detection of a rising edge on the SCL line. Although these bits are writable and readable, it is not necessary to access these bits under normal conditions.

To write to these bits, specify the number of bits to be transferred plus one (data is transferred with an additional acknowledge bit) between transferred frames. When setting any value other than $000_B$, set the value while the SCL line is at a low level.

[Clearing conditions]

- When 1 is written to the RIICnCR1.IICRST bit and a RIIC reset or internal reset is initiated.
- Data transfer including the acknowledge bit being completed.
- A start condition including a restart condition being detected.

## 20.3.5 RIICnMR2 — I2C Bus Mode Register 2

**Access:** RIICnMR2 register can be read or written in 32-bit units.
RIICnMR2L register can be read or written in 16-bit units.
RIICnMR2LL register can be read or written in 8-bit units.

**Address:** RIICnMR2: <RIICn_base> + $000C_H$

RIICnMR2L: <RIICn_base> + $000C_H$

RIICnMR2LL: <RIICn_base> + $000C_H$

**Value after reset:** $0000\ 0006_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | DLCS | SDDL[2:0] | | | — | TMOH | TMOL | TMOS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W |

**Table 20.13 RIICnMR2 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 7 | DLCS | SDA Output Delay Clock Source Selection<br>0: The internal reference clock (IICϕ) is selected as the clock source of the SDA output delay counter.<br>1: The internal reference clock divided by 2 (IICϕ/2) is selected as the clock source of the SDA output delay counter.*1 |
| 6 to 4 | SDDL[2:0] | SDA Output Delay Counter<br>• When RIICnMR2.DLCS = 0 (IICϕ)<br>b6 b4<br>0 0 0: No output delay<br>0 0 1: 1 IICϕ cycle<br>0 1 0: 2 IICϕ cycles<br>0 1 1: 3 IICϕ cycles<br>1 0 0: 4 IICϕ cycles<br>1 0 1: 5 IICϕ cycles<br>1 1 0: 6 IICϕ cycles<br>1 1 1: 7 IICϕ cycles<br>• When RIICnMR2.DLCS = 1 (IICϕ/2)<br>b6 b4<br>0 0 0: No output delay<br>0 0 1: 1 or 2 IICϕ cycles<br>0 1 0: 3 or 4 IICϕ cycles<br>0 1 1: 5 or 6 IICϕ cycles<br>1 0 0: 7 or 8 IICϕ cycles<br>1 0 1: 9 or 10 IICϕ cycles<br>1 1 0: 11 or 12 IICϕ cycles<br>1 1 1: 13 or 14 IICϕ cycles |
| 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 | TMOH | Timeout H Count Control<br>0: Disables counting while the SCL line is at a high level.<br>1: Enables counting while the SCL line is at a high level. |

Table 20.13 RIICnMR2 Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 1 | TMOL | Timeout L Count Control<br>0: Disables counting while the SCL line is at a low level.<br>1: Enables counting while the SCL line is at a low level. |
| 0 | TMOS | Timeout Detection Time Selection<br>0: Long mode is selected.<br>1: Short mode is selected. |

Note 1. The setting DLCS = 1 (IICϕ/2) only becomes valid when SCL is at the low level. When SCL is at the high level, the setting DLCS = 1 becomes invalid and the clock source becomes the internal reference clock (IICϕ).

### TMOS Bit (Timeout Detection Time Selection)

This bit is used to select long mode or short mode for the timeout detection time when the timeout function is enabled (RIICnFER.TMOE bit = 1). When this bit is set to 0, long mode is selected. When this bit is set to 1, short mode is selected. In long mode, the timeout detection internal counter functions as a 16 bit-counter. In short mode, the counter functions as a 14 bit-counter. While the SCL line is in the state that enables this counter as specified by bits TMOH and TMOL, the counter counts up in synchronization with the internal reference clock (IICϕ) as a count source.

For details on the timeout function, see **Section 20.13.1, Timeout Function**.

### TMOL Bit (Timeout L Count Control)

This bit is used to enable or disable the internal counter of the timeout function to count up while the SCL line is held low when the timeout function is enabled (RIICnFER.TMOE bit = 1).

### TMOH Bit (Timeout H Count Control)

This bit is used to enable or disable the internal counter of the timeout function to count up while the SCL line is held high when the timeout function is enabled (RIICnFER.TMOE bit = 1).

### SDDL[2:0] Bits (SDA Output Delay Setup Counter)

The SDA output can be delayed by the SDDL[2:0] setting. This counter works with the clock source selected by the DLCS bit. The setting of this function can be used for all types of SDA output, including the transmission of the acknowledge bit.

For details on this function, see **Section 20.7, Facility for Delaying SDA Output**.

**CAUTION**

Set the SDA output delay time to meet the I2C bus standard (within the data enable time/acknowledge enable time*1). Note that, if a value outside the standard is set, communication with communication devices may malfunction or it may seemingly become a start condition or stop condition depending on the bus state.

Note 1. Data enable time/acknowledge enable time
3,450 ns (up to 100 kbps: standard mode [Sm])
900 ns (up to 400 kbps: fast mode [Fm])

## 20.3.6 RIICnMR3 — I2C Bus Mode Register 3

**Access:** RIICnMR3 register can be read or written in 32-bit units.
RIICnMR3L register can be read or written in 16-bit units.
RIICnMR3LL register can be read or written in 8-bit units.

**Address:** RIICnMR3: <RIICn_base> + $0010_H$

RIICnMR3L: <RIICn_base> + $0010_H$

RIICnMR3LL: <RIICn_base> + $0010_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | WAIT | RDRFS | ACKWP | ACKBT | ACKBR | NF[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R | R/W | R/W |

**Table 20.14 RIICnMR3 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 6 | WAIT*2 | WAIT<br>0: No WAIT<br>(The period between ninth clock cycle and first clock cycle is not held low.)<br>1: WAIT<br>(The period between ninth clock cycle and first clock cycle is held low.)<br>Low-hold is released by reading RIICnDRR. |
| 5 | RDRFS*2 | RDRF Flag Set Timing Selection<br>0: The RDRF flag is set at the rising edge of the ninth SCL clock cycle.<br>(The SCL line is not held low at the falling edge of the eighth clock cycle.)<br>1: The RDRF flag is set at the rising edge of the eighth SCL clock cycle.<br>(The SCL line is held low at the falling edge of the eighth clock cycle.) Low-hold is released by writing a value to the ACKBT bit. |
| 4 | ACKWP*1 | ACKBT Write Protect<br>0: Modification of the ACKBT bit is disabled.<br>1: Modification of the ACKBT bit is enabled. |
| 3 | ACKBT*1 | Transmit Acknowledge<br>0: A 0 is sent as the acknowledge bit (ACK transmission).<br>1: A 1 is sent as the acknowledge bit (NACK transmission). |
| 2 | ACKBR | Receive Acknowledge<br>0: A 0 is received as the acknowledge bit (ACK reception).<br>1: A 1 is received as the acknowledge bit (NACK reception). |
| 1, 0 | NF[1:0] | Digital noise Filter Stage Selection<br>b1 b0<br>0 0: Noise of up to one IICϕ cycle is filtered out (single-stage filter).<br>0 1: Noise of up to two IICϕ cycles is filtered out (2-stage filter).<br>1 0: Noise of up to three IICϕ cycles is filtered out (3-stage filter).<br>1 1: Noise of up to four IICϕ cycles is filtered out (4-stage filter). |

Note 1. If it is attempted to write 1 to both ACKWP and ACKBT bits, the ACKBT bit cannot be set to 1.

Note 2. The WAIT and RDRFS bits are valid only in receive mode (invalid in transmit mode).

### NF[1:0] Bits (Digital noise Filter Stage Selection)

These bits are used to select the number of stages of the digital noise filter.

**CAUTION**

Set the noise range to be filtered out by the noise filter within a range less than the SCL line high-level period or low-level period. If the noise range is set to a value of (SCL clock width: high-level period or low-level period, whichever is shorter) - [1.5 internal reference clock synchronized (IICϕ) cycles] or more, the SCL clock is regarded as noise by the noise filter function of the RIIC, which may prevent the RIIC from operating normally.

### ACKBR Bit (Receive Acknowledge)

This bit is used to store the acknowledge bit information received from the receive device in transmit mode.

[Setting condition]

When 1 is received as the acknowledge bit with the RIICnCR2.TRS bit set to 1

[Clearing conditions]

- When 0 is received as the acknowledge bit with the RIICnCR2.TRS bit set to 1
- When 1 is written to the RIICnCR1.IICRST bit while the RIICnCR1.ICE bit is 0 (RIIC reset)

### ACKBT Bit (Transmit Acknowledge)

This bit is used to set the bit to be sent at the acknowledge timing in receive mode.

[Setting condition]

When 1 is written to this bit with the ACKWP bit set to 1

[Clearing conditions]

- When 0 is written to this bit after ACKWP reading while the ACKWP bit is set to 1
- When a stop condition is detected
- When 1 is written to the RIICnCR1.IICRST bit while the RIICnCR1.ICE bit is 0 (RIIC reset)

**CAUTION**

The ACKBT bit must be written to while the ACKWP bit is 1. If the ACKBT bit is written to with the ACKWP bit cleared to 0, writing to the ACKBT bit is disabled.

### ACKWP Bit (ACKBT Write Protect)

This bit is used to control the modification of the ACKBT bit.

### RDRFS Bit (RDRF Flag Set Timing Selection)

This bit is used to select the RDRF flag set timing in receive mode and also to select whether to hold the SCL line low at the falling edge of the eighth SCL clock cycle.

When the RDRFS bit is 0, the SCL line is not held low at the falling edge of the eighth SCL clock cycle, and the RDRF flag is set to 1 at the rising edge of the ninth SCL clock cycle.

When the RDRFS bit is 1, the RDRF flag is set to 1 at the rising edge of the eighth SCL clock cycle and the SCL line is held low at the falling edge of the eighth SCL clock cycle. The low-hold of the SCL line is released by writing a value to the ACKBT bit.

After data is received with this setting, the SCL line is automatically held low before the acknowledge bit is sent. This enables processing to send ACK (ACKBT = 0) or NACK (ACKBT = 1) according to receive data.

### WAIT Bit (WAIT)

This bit is used to control whether to hold the period between the ninth SCL clock cycle and the first SCL clock cycle low until the receive data buffer (RIICnDRR) is completely read each time single-byte data is received in receive mode.

When the WAIT bit is 0, the receive operation is continued without holding the period between the ninth and the first SCL clock cycle low. When both the RDRFS and WAIT bits are 0, continuous receive operation is enabled with the double buffer.

When the WAIT bit is 1, the SCL line is held low from the falling edge of the ninth clock cycle until the RIICnDRR value is read each time single-byte data is received. This enables receive operation in byte units.

**CAUTION**

When the value of the WAIT bit is cleared to 0, be sure to read the RIICnDRR beforehand.

## 20.3.7 RIICnFER — I2C Bus Function Enable Register

**Access:** RIICnFER register can be read or written in 32-bit units.
RIICnFERL register can be read or written in 16-bit units.
RIICnFERLL register can be read or written in 8-bit units.

**Address:** RIICnFER: <RIICn_base> + $0014_H$

RIICnFERL: <RIICn_base> + $0014_H$

RIICnFERLL: <RIICn_base> + $0014_H$

**Value after reset:** $0000\ 0072_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | SCLE | NFE | NACKE | SALE | NALE | MALE | TMOE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.15 RIICnFER Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 6 | SCLE | SCL Synchronous Circuit Enable<br>0: No SCL synchronous circuit is used.<br>1: An SCL synchronous circuit is used. |
| 5 | NFE | Digital Noise Filter Circuit Enable<br>0: No digital noise filter circuit is used.<br>1: A digital noise filter circuit is used. |
| 4 | NACKE | NACK Reception Transfer Suspension Enable<br>0: Transfer operation is not suspended during NACK reception (transfer suspension disabled).<br>1: Transfer operation is suspended during NACK reception (transfer suspension enabled). |
| 3 | SALE | Slave Arbitration-Lost Detection Enable<br>0: Slave arbitration-lost detection is disabled.<br>1: Slave arbitration-lost detection is enabled. |
| 2 | NALE | NACK Transmission Arbitration-Lost Detection Enable<br>0: NACK transmission arbitration-lost detection is disabled.<br>1: NACK transmission arbitration-lost detection is enabled. |
| 1 | MALE | Master Arbitration-Lost Detection Enable<br>0: Master arbitration-lost detection is disabled.<br>(Disables the arbitration-lost detection function and does not clear the RIICnCR2.MST and TRS bits automatically when arbitration is lost.)<br>1: Master arbitration-lost detection is enabled.<br>(Enables the arbitration-lost detection function and clears the RIICnCR2.MST and TRS bits automatically when arbitration is lost.)<br><br>If 1 is written to the ST bit while the BBSY flag is 1 in slave transmit mode, the TRS bit is not cleared. |
| 0 | TMOE | Timeout Function Enable<br>0: The timeout function is disabled.<br>1: The timeout function is enabled. |

#### TMOE Bit (Timeout Function Enable)

This bit is used to enable or disable the timeout function.

For details on the timeout function, see **Section 20.13.1, Timeout Function**.

#### MALE Bit (Master Arbitration-Lost Detection Enable)

This bit is used to select enabling or disabling of the arbitration-lost detection function. Normally, set this bit to 1.

#### NALE Bit (NACK Transmission Arbitration-Lost Detection Enable)

This bit is used to specify whether the detection of ACK during transmission of NACK in reception (such as when slaves with the same address are present on the bus and each is transmitting different data, or when two or more masters select the same slave device simultaneously with different numbers of bytes for reception) is judged to represent a loss in arbitration.

#### SALE Bit (Slave Arbitration-Lost Detection Enable)

This bit is used to specify whether to cause arbitration to be lost when a value different from the value being transmitted is detected on the bus in slave transmit mode (such as when slaves with the same address exist on the bus or when a mismatch with the transmit data occurs due to noise).

#### NACKE Bit (NACK Reception Transfer Suspension Enable)

This bit is used to specify whether to continue or discontinue the transfer operation when NACK is received in transmit mode. Normally, set this bit to 1.

When NACK is received with the NACKE bit set to 1, the next transfer operation is suspended.

When the NACKE bit is 0, the next transfer operation is continued regardless of the received acknowledge content.

#### SCLE Bit (SCL Synchronous Circuit Enable)

This bit is used to specify whether to synchronize the SCL clock with a rising or falling edge on the SCL line. Normally, set this bit to 1.

When the SCLE bit is cleared to 0 (SCL synchronous circuit not used), the RIIC does not synchronize the SCL clock with the SCL input clock. In this setting, the RIIC outputs the SCL clock with the transfer rate set in RIICnBRH and RIICnBRL regardless of the SCL line state. For this reason, if the bus load of the I2C bus line is much larger than the specification value or if the SCL clock output overlaps in multiple masters, the short-cycle SCL clock that does not meet the specification may be output. When no SCL synchronous circuit is used, it also affects the issuance of a start condition, restart condition, and stop condition, and the continuous output of extra SCL clock cycles.

This bit must not be cleared to 0 except for checking the output of the transfer rate.

## 20.3.8 RIICnSER — I2C Bus Status Enable Register

**Access:** RIICnSER register can be read or written in 32-bit units.
RIICnSERL register can be read or written in 16-bit units.
RIICnSERLL register can be read or written in 8-bit units.

**Address:** RIICnSER: <RIICn_base> + $0018_H$

RIICnSERL: <RIICn_base> + $0018_H$

RIICnSERLL: <RIICn_base> + $0018_H$

**Value after reset:** $0000\ 0009_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | DIDE | — | GCE | SAR2 | SAR1 | SAR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R | R/W | R/W | R/W | R/W |

**Table 20.16 RIICnSER Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5 | DIDE | Device-ID Address Detection Enable<br>0: Device-ID address detection is disabled.<br>1: Device-ID address detection is enabled. |
| 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | GCE | General Call Address Enable<br>0: General call address detection is disabled.<br>1: General call address detection is enabled. |
| 2 | SAR2 | Slave Address Register 2 Enable<br>0: Slave address in RIICnSAR2 is disabled.<br>1: Slave address in RIICnSAR2 is enabled. |
| 1 | SAR1 | Slave Address Register 1 Enable<br>0: Slave address in RIICnSAR1 is disabled.<br>1: Slave address in RIICnSAR1 is enabled. |
| 0 | SAR0 | Slave Address Register 0 Enable<br>0: Slave address in RIICnSAR0 is disabled.<br>1: Slave address in RIICnSAR0 is enabled. |

#### SARy Bit (Slave Address Register y Enable) (y = 0 to 2)

This bit is used to enable or disable the slave address set in RIICnSARy.

When this bit is set to 1, the slave address set in RIICnSARy is enabled and is compared with the received slave address.

When this bit is cleared to 0, the slave address set in RIICnSARy is disabled and is ignored even if it matches the received slave address.

#### GCAE Bit (General Call Address Enable)

This bit is used to specify whether to ignore the general call address ($0000\ 000_B$ + 0 [W]: All 0) when it is received.

When this bit is set to 1, if the received slave address matches the general call address, the RIIC recognizes the received slave address as the general call address independently of the slave addresses set in RIICnSARy (y = 0 to 2) and performs data receive operation.

When this bit is cleared to 0, the received slave address is ignored even if it matches the general call address.

#### DIDE Bit (Device-ID Address Detection Enable)

This bit is used to specify whether to recognize and execute the Device-ID address when a device ID ($1111\ 100_B$) is received in the first frame after a start condition or restart condition is detected.

When this bit is set to 1, if the received first frame matches the device ID, the RIIC recognizes that the Device-ID address has been received. When the following R/W# bit is 0 [W], the RIIC recognizes the second and the following frames as slave addresses and continues the receive operation.

When this bit is cleared to 0, the RIIC ignores the received first frame even if it matches the device ID address and recognizes the first frame as a normal slave address.

For details on the device-ID address detection, see **Section 20.9.3, Device-ID Address Detection**.

## 20.3.9 RIICnIER — I2C Bus Interrupt Enable Register

**Access:** RIICnIER register can be read or written in 32-bit units.
RIICnIERL register can be read or written in 16-bit units.
RIICnIERLL register can be read or written in 8-bit units.

**Address:** RIICnIER: <RIICn_base> + $001C_H$

RIICnIERL: <RIICn_base> + $001C_H$

RIICnIERLL: <RIICn_base> + $001C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TIE | TEIE | RIE | NAKIE | SPIE | STIE | ALIE | TMOIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.17 RIICnIER Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 7 | TIE | Transmit Data Empty Interrupt Enable<br>0: Transmit data empty interrupt request (INTRIICnTI) is disabled.<br>1: Transmit data empty interrupt request (INTRIICnTI) is enabled. |
| 6 | TEIE | Transmit End Interrupt Enable<br>0: Transmit end interrupt request (INTRIICnTEI) is disabled.<br>1: Transmit end interrupt request (INTRIICnTEI) is enabled. |
| 5 | RIE | Receive Complete Interrupt Enable<br>0: Receive complete interrupt request (INTRIICnRI) is disabled.<br>1: Receive complete interrupt request (INTRIICnRI) is enabled. |
| 4 | NAKIE | NACK Reception Interrupt Enable<br>0: NACK reception interrupt request (NAKI) is disabled.<br>1: NACK reception interrupt request (NAKI) is enabled. |
| 3 | SPIE | Stop Condition Detection Interrupt Enable<br>0: Stop condition detection interrupt request (SPI) is disabled.<br>1: Stop condition detection interrupt request (SPI) is enabled. |
| 2 | STIE | Start Condition Detection Interrupt Enable<br>0: Start condition detection interrupt request (STI) is disabled.<br>1: Start condition detection interrupt request (STI) is enabled. |
| 1 | ALIE | Arbitration-Lost Interrupt Enable<br>0: Arbitration-lost interrupt request (ALI) is disabled.<br>1: Arbitration-lost interrupt request (ALI) is enabled. |
| 0 | TMOIE | Timeout Interrupt Enable<br>0: Timeout interrupt request (TMOI) is disabled.<br>1: Timeout interrupt request (TMOI) is enabled. |

### TMOIE Bit (Timeout Interrupt Enable)

This bit is used to enable or disable timeout interrupt requests (TMOI) when the RIICnSR2.TMOF flag is set to 1. A TMOI interrupt request is canceled by clearing the TMOF flag or the TMOIE bit to 0.

#### ALIE Bit (Arbitration-Lost Interrupt Enable)

This bit is used to enable or disable arbitration-lost interrupt requests (ALI) when the RIICnSR2.AL flag is set to 1. An ALI interrupt request is canceled by clearing the AL flag or the ALIE bit to 0.

#### STIE Bit (Start Condition Detection Interrupt Enable)

This bit is used to enable or disable start condition detection interrupt requests (STI) when the RIICnSR2.START flag is set to 1. An STI interrupt request is canceled by clearing the START flag or the STIE bit to 0.

#### SPIE Bit (Stop Condition Detection Interrupt Enable)

This bit is used to enable or disable stop condition detection interrupt requests (SPI) when the RIICnSR2.STOP flag is set to 1. An SPI interrupt request is canceled by clearing the STOP flag or the SPIE bit to 0.

#### NAKIE Bit (NACK Reception Interrupt Enable)

This bit is used to enable or disable NACK reception interrupt requests (NAKI) when the RIICnSR2.NACKF flag is set to 1. An NAKI interrupt request is canceled by clearing the NACKF flag or the NAKIE bit to 0.

#### RIE Bit (Receive Complete Interrupt Enable)

This bit is used to enable or disable receive complete interrupt requests (INTRIICnRI) when the RIICnSR2.RDRF flag is set to 1.

#### TEIE Bit (Transmit End Interrupt Enable)

This bit is used to enable or disable transmit end interrupts (INTRIICnTEI) when the RIICnSR2.TEND flag is set to 1. An INTIICnTEI interrupt request is canceled by clearing the TEND flag or the TEIE bit to 0.

#### TIE Bit (Transmit Data Empty Interrupt Enable)

This bit is used to enable or disable transmit data empty interrupts (INTRIICnTI) when the RIICnSR2.TDRE flag is set to 1.

## 20.3.10 RIICnSR1 — I2C Bus Status Register 1

**Access:** RIICnSR1 register can be read or written in 32-bit units.
RIICnSR1L register can be read or written in 16-bit units.
RIICnSR1LL register can be read or written in 8-bit units.

**Address:** RIICnSR1: <RIICn_base> + $0020_H$

RIICnSR1L: <RIICn_base> + $0020_H$

RIICnSR1LL: <RIICn_base> + $0020_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | DID | — | GCA | AAS2 | AAS1 | AAS0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R(/W) *1 | R | R(/W) *1 | R(/W) *1 | R(/W) *1 | R(/W) *1 |

Note 1. Only 0 can be written to this bit.

**Table 20.18 RIICnSR1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5 | DID | Device-ID Address Detection Flag<br>0: Device-ID address is not detected.<br>1: Device-ID address is detected. |
| 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | GCA | General Call Address Detection Flag<br>0: General call address is not detected.<br>1: General call address is detected. |
| 2 | AAS2 | Slave Address 2 Detection Flag<br>0: Slave address 2 is not detected.<br>1: Slave address 2 is detected. |
| 1 | AAS1 | Slave Address 1 Detection Flag<br>0: Slave address 1 is not detected.<br>1: Slave address 1 is detected. |
| 0 | AAS0 | Slave Address 0 Detection Flag<br>0: Slave address 0 is not detected.<br>1: Slave address 0 is detected. |

### AASy Flag (Slave Address y Detection) (y = 0 to 2)

[Setting conditions]

<For 7-bit address format: RIICnSARy.FSy = 0>

When the received slave address matches the RIICnSARy.SVA[7:1] value with the RIICnSER.SARy bit set to 1 (slave address y detection enabled)
This flag is set to 1 at the rising edge of the ninth SCL clock cycle in the frame.

<For 10-bit address format: RIICnSARy.FSy = 1>

When the received slave address matches a value of (1111 $0_B$ + RIICnSARy.SVA[9:8]) and the following address matches the RIICnSARy.SVA[7:0] value with the RIICnSER.SARy bit set to 1 (slave address y detection enabled)
This flag is set to 1 at the rising edge of the ninth SCL clock cycle in the frame.

[Clearing conditions]

- When 0 is written to the AASy bit after reading AASy = 1
- When a stop condition is detected
- When 1 is written to the RIICnCR1.IICRST bit to apply an RIIC reset or an internal reset

<For 7-bit address format: RIICnSARy.FSy = 0>

- When the received slave address does not match the RIICnSARy.SVA[7:1] value with the RIICnSER.SARy bit set to 1 (slave address y detection enabled)
  This flag is cleared to 0 at the rising edge of the ninth SCL clock cycle in the frame.

<For 10-bit address format: RIICnSARy.FSy = 1>

- When the received slave address does not match a value of (1111 $0_B$ + RIICnSARy.SVA[9:8]) with the RIICnSER.SARy bit set to 1 (slave address y detection enabled)
  This flag is cleared to 0 at the rising edge of the ninth SCL clock cycle in the frame.
- When the received slave address matches a value of (1111 $0_B$ + RIICnSARy.SVA[9:8]) and the following address does not match the RIICnSARy.SVA[7:0] value with the RIICnSER.SARy bit set to 1 (slave address y detection enabled)
  This flag is cleared to 0 at the rising edge of the ninth SCL clock cycle in the frame.

### GCA Flag (General Call Address Detection)

[Setting condition]

- When the received slave address matches the general call address (0000 $000_B$ + 0 [W]) with the RIICnSER.GCE bit set to 1 (general call address detection enabled)
  This flag is set to 1 at the rising edge of the ninth SCL clock cycle in the frame.

[Clearing conditions]

- When 0 is written to the GCA bit after reading GCA = 1
- When a stop condition is detected
- When the received slave address does not match the general call address (0000 $000_B$ + 0 [W]) with the RIICnSER.GCE bit set to 1 (general call address detection enabled)
  This flag is cleared to 0 at the rising edge of the ninth SCL clock cycle in the frame.
- When 1 is written to the RIICnCR1.IICRST bit to apply an RIIC reset or an internal reset

### DID Flag (Device-ID Address Detection)

[Setting condition]

- When the first frame received immediately after a start condition or restart condition is detected matches a value of (device ID ($1111\ 100_B$) + 0 [W]) with the RIICnSER.DIDE bit set to 1 (Device-ID address detection enabled). This flag is set to 1 at the rising edge of the ninth SCL clock cycle in the frame.
- When a restart condition is detected after a match with the device ID address and the device ID address ($1111\ 100_B$) plus 1[R] has matched while the setting of the RIICnSER.DIDE bit is 1 (device ID address detection enabled). This flag is set to 1 at the rising edge of the ninth SCL clock cycle in the frame.

[Clearing conditions]

- When 0 is written to the DID bit after reading DID = 1
- When a stop condition is detected
- When the first frame received immediately after a start condition or restart condition is detected does not match a value of (device ID ($1111\ 100_B$)) with the RIICnSER.DIDE bit set to 1 (Device-ID address detection enabled)
  This flag is cleared to 0 at the rising edge of the ninth SCL clock cycle in the frame.
- When the first frame received immediately after a start condition or restart condition is detected matches a value of (device ID ($1111\ 100_B$) + 0 [W]) and the second frame does not match any of slave addresses 0 to 2 with the RIICnSER.DIDE bit set to 1 (Device-ID address detection enabled)
  This flag is cleared to 0 at the rising edge of the ninth SCL clock cycle in the frame.
- When 1 is written to the RIICnCR1.IICRST bit to apply an RIIC reset or an internal reset

## 20.3.11 RIICnSR2 — I2C Bus Status Register 2

**Access:** RIICnSR2 register can be read or written in 32-bit units.
RIICnSR2L register can be read or written in 16-bit units.
RIICnSR2LL register can be read or written in 8-bit units.

**Address:** RIICnSR2: <RIICn_base> + $0024_H$

RIICnSR2L: <RIICn_base> + $0024_H$

RIICnSR2LL: <RIICn_base> + $0024_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TDRE | TEND | RDRF | NACKF | STOP | START | AL | TMOF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R(/W) *1 | R(/W) *1 | R(/W) *1 | R(/W) *1 | R(/W) *1 | R(/W) *1 | R(/W) *1 |

Note 1. Only 0 can be written to this bit.

**Table 20.19 RIICnSR2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 7 | TDRE | Transmit Data Empty Flag<br>0: RIICnDRT contains transmit data.<br>1: RIICnDRT contains no transmit data. |
| 6 | TEND | Transmit End Flag<br>0: Data is being transmitted.<br>1: Data has been transmitted. |
| 5 | RDRF | Receive Complete Flag<br>0: RIICnDRR contains no receive data.<br>1: RIICnDRR contains receive data. |
| 4 | NACKF | NACK Detection Flag<br>0: NACK is not detected.<br>1: NACK is detected. |
| 3 | STOP | Stop Condition Detection Flag<br>0: Stop condition is not detected.<br>1: Stop condition is detected. |
| 2 | START | Start Condition Detection Flag<br>0: Start condition is not detected.<br>1: Start condition is detected. |
| 1 | AL | Arbitration-Lost Flag<br>0: Arbitration is not lost.<br>1: Arbitration is lost. |
| 0 | TMOF | Timeout Detection Flag<br>0: Timeout is not detected.<br>1: Timeout is detected. |

### TMOF Flag (Timeout Detection)

This flag is set to 1 when the RIIC recognizes timeout after the SCL line state remains unchanged for a certain period.

[Setting condition]

The timeout function is enabled when the RIICnFER.TMOE bit is 1. It detects an abnormal bus state that the SCL line is stuck low or high during the following conditions:

- The bus is busy (the RIICnCR2.BBSY flag is 1) in master mode (the RIICnCR2.MST bit is 1).
- The RIIC's own slave address matches (the RIICnSR1 register is not $00_H$) and the bus is busy (the RIICnCR2.BBSY bit is 1) in slave mode (the RIICnCR2.MST bit is 0).
- The bus is free (the RIICnCR2.BBSY flag is 0) while generation of a start condition is requested (the RIICnCR2.ST bit is 1).

[Clearing conditions]

- When 0 is written to the TMOF bit after reading TMOF = 1
- When 1 is written to the RIICnCR1.IICRST bit to apply an RIIC reset or an internal reset

### AL Flag (Arbitration-Lost)

This flag shows that bus mastership has been lost (loss in arbitration) due to a bus conflict or some other reason when a start condition is issued or an address and data are transmitted. The RIIC monitors the level on the SDA line during transmission and, if the level on the line does not match the value of the bit being output, sets the value of the AL bit to 1 to indicate that the bus is occupied by another device.

The RIIC can also set the flag to indicate the detection of loss of arbitration during NACK transmission in receive mode or during data transmission in slave mode.

[Setting conditions]

<When master arbitration-lost detection is enabled: RIICnFER.MALE = 1>

- When the internal SDA output state does not match the SDA line level at the rising edge of SCL clock except for the ACK period during data (including slave address) transmission in master transmit mode (when the SDA line is driven low while the internal SDA output is at a high level (the SDA pin is in the high-impedance state))
- When a start condition is detected while the RIICnCR2.ST bit is 1 (start condition issuance request) or the internal SDA output state does not match the SDA line level
- When the RIICnCR2.ST bit is set to 1 (start condition issuance request) with the RIICnCR2.BBSY flag set to 1.

<When NACK arbitration-lost detection is enabled: RIICnFER.NALE = 1>

When the internal SDA output state does not match the SDA line level at the rising edge of SCL clock in the ACK period during NACK transmission in receive mode

<When slave arbitration-lost detection is enabled: RIICnFER.SALE = 1>

When the internal SDA output state does not match the SDA line level at the rising edge of SCL clock except for the ACK period during data transmission in slave transmit mode

[Clearing conditions]

- When 0 is written to the AL bit after reading AL = 1

- When 1 is written to the RIICnCR1.IICRST bit to apply an RIIC reset or an internal reset

**Table 20.20 Relationship between Arbitration-Lost Generation Sources and Arbitration-Lost Enable Functions**

| RIICnFER | | | RIICnSR2 | | |
|---|---|---|---|---|---|
| MALE | NALE | SALE | AL | Error | Arbitration-Lost Generation Source |
| 1 | — | — | 1 | Start condition issuance error | When internal SDA output state does not match SDA line level when a start condition is detected while the RIICnCR2.ST bit is 1 |
| | | | | | When RIICnCR2.ST is set to 1 with RIICnCR2.BBSY set to 1 |
| | | | 1 | Transmit data mismatch | When transmit data (including slave address) does not match the bus state in master transmit mode |
| — | 1 | — | 1 | NACK transmission mismatch | When ACK is detected during transmission of NACK in master receive mode or slave receive mode |
| — | — | 1 | 1 | Transmit data mismatch | When transmit data does not match the bus state in slave transmit mode |

—: Don't care

### START Flag (Start Condition Detection)

[Setting condition]

When a start condition (or a restart condition) is detected

[Clearing conditions]

- When 0 is written to the START bit after reading START = 1
- When a stop condition is detected
- When 1 is written to the RIICnCR1.IICRST bit to apply an RIIC reset or an internal reset

### STOP Flag (Stop Condition Detection)

[Setting condition]

When a stop condition is detected

[Clearing conditions]

- When 0 is written to the STOP bit after reading STOP = 1
- When 1 is written to the RIICnCR1.IICRST bit to apply an RIIC reset or an internal reset

### NACKF Flag (NACK Detection)

[Setting condition]

When acknowledge is not received (NACK is received) from the receive device in transmit mode with the RIICnFER.NACKE bit set to 1 (transfer suspension enabled)

[Clearing conditions]

- When 0 is written to the NACKF bit after reading NACKF = 1
- When 1 is written to the RIICnCR1.IICRST bit to apply an RIIC reset or an internal reset

**CAUTION**

When the NACKF flag is set to 1, the RIIC suspends data transmission/reception. Writing to RIICnDRT in transmit mode or reading from RIICnDRR in receive mode with the NACKF flag set to 1 does not enable data transmit/receive operation. To restart data transmission/reception, clear the NACKF flag to 0.

### RDRF Flag (Receive Complete)

[Setting conditions]

- When receive data has been transferred from RIICnDRS to RIICnDRR
  At the rising edge of the eighth or ninth SCL clock cycle (selected by the RIICnMR3.RDRFS bit)
- When the received slave address matches the address enabled in RIICnSER after a start condition (or a restart condition) is detected with the RIICnCR2.TRS bit cleared to 0
- In master mode, transition to master reception while the R/W# bit appended to the slave address is set to 1

[Clearing conditions]

- When 0 is written to the RDRF bit after reading RDRF = 1
- When data is read from RIICnDRR
- When 1 is written to the RIICnCR1.IICRST bit to apply an RIIC reset or an internal reset

### TEND Flag (Transmit End)

[Setting condition]

At the rising edge of the ninth SCL clock cycle while the TDRE flag is 1

[Clearing conditions]

- When 0 is written to the TEND bit after reading TEND = 1
- When data is written to RIICnDRT
- When a stop condition is detected
- When 1 is written to the RIICnCR1.IICRST bit to apply an RIIC reset or an internal reset

### TDRE Flag (Transmit Data Empty)

[Setting conditions]

- When data has been transferred from RIICnDRT to RIICnDRS and RIICnDRT becomes empty
- When the RIICnCR2.TRS bit is set to 1
  - When the RIICnCR2.MST bit is set to 1 after a start condition is detected
  - When 1 is written to the RIICnCR2.TRS bit while the RIICnMR1.MTWP bit is 1
- When the received slave address matches the address enabled in RIICnSER after a start condition including a restart condition is detected with the RIICnCR2.TRS bit set to 1

[Clearing conditions]

- When data is written to RIICnDRT
- When the RIICnCR2.TRS bit is cleared to 0
  - When a stop condition is detected
  - When the RIIC enters receive mode from transmit mode
  - When 0 is written to the RIICnCR2.TRS bit while the RIICnMR1.MTWP bit is 1
- When 1 is written to the RIICnCR1.IICRST bit to apply an RIIC reset or an internal reset

**CAUTION**

When the NACKF flag is set to 1 while the RIICnFER.NACKE bit is 1, the RIIC suspends data transmission/reception. Here, if the TDRE flag is 0 (next transmit data has been written), data is transferred to the RIICnDRS register and the RIICnDRT register becomes empty at the rising edge of the ninth clock cycle, but the TDRE flag is not set to 1.

## 20.3.12 RIICnSARy — I2C Slave Address Register y (y = 0 to 2)

**Access:** RIICnSARy register can be read or written in 32-bit units.
RIICnSARyL register can be read or written in 16-bit units.
RIICnSARyLL and RIICnSARyLH registers can be read or written in 8-bit units.

**Address:** RIICnSAR0: <RIICn_base> + $0028_H$

RIICnSAR0L: <RIICn_base> + $0028_H$

RIICnSAR0LL: <RIICn_base> + $0028_H$,
RIICnSAR0LH: <RIICn_base> + $0029_H$

RIICnSAR1: <RIICn_base> + $002C_H$

RIICnSAR1L: <RIICn_base> + $002C_H$

RIICnSAR1LL: <RIICn_base> + $002C_H$,
RIICnSAR1LH: <RIICn_base> + $002D_H$

RIICnSAR2: <RIICn_base> + $0030_H$

RIICnSAR2L: <RIICn_base> + $0030_H$

RIICnSAR2LL: <RIICn_base> + $0030_H$,
RIICnSAR2LH: <RIICn_base> + $0031_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FSy | — | — | — | — | — | SVA[9:1] | | | | | | | | | SVA0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.21 RIICnSARy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 15 | FSy | 7-Bit/10-Bit Address Format Selection<br>0: The 7-bit address format is selected.<br>1: The 10-bit address format is selected. |
| 14 to 10 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 9 to 1 | SVA[9:1] | 7-Bit Address/10-Bit Address Upper Bits<br>A slave address is set.<br>• When the FSy bit is 0 (7-bit address format), the SVA[7:1] bits are valid and form a 7-bit slave address.<br>• When the FSy bit is 1 (10-bit address format), SVA[9:1] bits form a 10-bit slave address (combined with the SVA0 bit). |
| 0 | SVA0 | 10-Bit Address LSB<br>The least significant bit (LSB) of a 10-bit slave address is set.<br>• When the FSy bit is 0 (7-bit address format), this bit is invalid.<br>• When the FSy bit is 1 (10-bit address format), this bit is a 10-bit slave address (combined with the SVA[9:1] bits). |

#### SVA0 Bit (10-Bit Address LSB)

When the 10-bit address format is selected (RIICnSARy.FSy = 1), this bit functions as the LSB of a 10-bit address and forms a 10-bit address in combination with the SVA[9:1] bits.

When the RIICnSER.SARy bit is set to 1 (RIICnSARy enabled) and the RIICnSARy.FSy bit is 1, this bit is valid. While the RIICnSARy.FSy bit or SARy bit is 0, the setting of this bit is ignored.

#### SVA[9:1] Bits (7-Bit Address/10-Bit Address Upper Bits)

When the 7-bit address format is selected (RIICnSARy.FSy = 0), these bits function as a 7-bit address. When the 10-bit address format is selected (RIICnSARy.FSy = 1), these bits function as a 10-bit address in combination with the SVA0 bit.

While the RIICnSER.SARy bit is 0, the setting of these bits is ignored.

#### FSy Bit (7-Bit/10-Bit Address Format Selection)

This bit is used to select 7-bit address or 10-bit address for slave address y (in RIICnSARy).

When the RIICnSER.SARy bit is set to 1 (RIICnSARy enabled) and the RIICnSARy.FSy bit is 0, the 7-bit address format is selected for slave address y, the RIICnSARy.SVA[7:1] setting is valid, and the settings of the SVA[9:8] bits and the RIICnSARy.SVA0 bit are ignored.

When the RIICnSER.SARy bit is set to 1 (RIICnSARy enabled) and the RIICnSARy.FSy bit is 1, the 10-bit address format is selected for slave address y and the settings of the SVA[9:1] bits and the SVA0 bit are valid.

While the RIICnSER.SARy bit is 0 (RIICnSARy disabled), the setting of the RIICnSARy.FSy bit is invalid.

## 20.3.13 RIICnBRL — I2C Bus Bit Rate Low-Level Register

**Access:** RIICnBRL register can be read or written in 32-bit units.
RIICnBRLL register can be read or written in 16-bit units.
RIICnBRLLL register can be read or written in 8-bit units.

**Address:** RIICnBRL: <RIICn_base> + $0034_H$

RIICnBRLL: <RIICn_base> + $0034_H$

RIICnBRLLL: <RIICn_base> + $0034_H$

**Value after reset:** $0000\ 00FF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | BRL[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 20.22 RIICnBRL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 4 to 0 | BRL[4:0] | Bit Rate Low-Level Period<br>Low-level period of SCL clock |

The RIICnBRL register is a 5-bit register that is used to set the width at low level for the SCL clock.

It also works to generate the data setup time for automatic SCL low-hold operation (see **Section 20.10, Automatic Low-Hold Function for SCL**); when the RIIC is used only in slave mode, this register needs to be set to a value longer than the data setup time*1.

RIICnBRL counts the low-level period with the internal reference clock source (IICϕ) specified by the RIICnMR1.CKS[2:0] bits.

Note 1. Data setup time ($t_{SU}$: DAT)
250 ns (up to 100 kbps: standard mode [Sm])
100 ns (up to 400 kbps: fast mode [Fm])

## 20.3.14 RIICnBRH — I2C Bus Bit Rate High-Level Register

**Access:** RIICnBRH register can be read or written in 32-bit units.
RIICnBRHL register can be read or written in 16-bit units.
RIICnBRHLL register can be read or written in 8-bit units.

**Address:** RIICnBRH: <RIICn_base> + $0038_H$

RIICnBRHL: <RIICn_base> + $0038_H$

RIICnBRHLL: <RIICn_base> + $0038_H$

**Value after reset:** $0000\ 00FF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | BRH[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 20.23 RIICnBRH Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 4 to 0 | BRH[4:0] | Bit Rate High-Level Period<br>High-level period of SCL clock |

RIICnBRH is a 5-bit register to set the high-level period of SCL clock. RIICnICBRH is valid in master mode. If the RIIC is used only in slave mode, this register need not to set the high-level period.

RIICnBRH counts the high-level period with the internal reference clock source (IICϕ) specified by the RIICnMR1.CKS[2:0] bits.

The I2C transfer rate and the SCL clock duty are calculated using the following expression.

(1) When RIICnFER.SCLE = 0
Transfer rate = 1 / {[ (RIICnBRH+1) + (RIICnBRL+1) ] / IICφ*1 + tr + tf}
Duty cycle = {tr+[(RIICnBRH+1)/ IICφ]}/{ tr + tf + [(RIICnBRH+1) + (RIICnBRL+1)]/IICφ}

(2) When RIICnFER.SCLE=1, RIICnFER.NFE=0, IICφ = PCLK
Transfer rate = 1 / {[ (RIICnBRH+3) + (RIICnBRL+3) ] / IICφ*1 + tr + tf}
Duty cycle = {tr +[(RIICnBRH+3)/ IICφ]}/{ tr + tf + [(RIICnBRH+3) + (RIICnBRL+3)]/IICφ}

(3) When RIICnFER.SCLE=1, RIICnFER.NFE=1, IICφ = PCLK
Transfer rate =1 / {[ (RIICnBRH+3+nf) + (RIICnBRL+3+nf) ] / IICφ*1 + tr + tf}
Duty cycle = {tr +[(RIICnBRH+3+nf)/ IICφ]}/{ tr + tf + [(RIICnBRH+3+nf) + (RIICnBRL+3+nf)]/ IICφ}

(4) RIICnFER.SCLE=1, RIICnFER.NFE=0, IICφ < PCLK
Transfer rate =1 / {[ (RIICnBRH+2) + (RIICnBRL+2) ] / IICφ*1+ tr + tf}
Duty cycle = {tr +[(RIICnBRH+2)/ IICφ]}/{ tr + tf + [(RIICnBRH+2) + (RIICnBRL+2)]/IICφ}

(5) When RIICnFER.SCLE=1, RIICnFER.NFE=1, IICφ < PCLK
Transfer rate =1 / {[ (RIICnBRH+2+nf) + (RIICnBRL+2+nf) ] / IICφ*1+ tr + tf}
Duty cycle = {tr +[(RIICnBRH+2+nf)/ IICφ]}/{ tr + tf + [(RIICnBRH+2+nf) + (RIICnBRL+2+nf)]/ IICφ}

tf: SCL line falling time [ns]*2
tr: SCL line rising time [ns]*2
nf: Digital noise filter stage
Duty cycle: 0% < Duty < 100%

**Note 1.** As for IICφ, see CKS[2:0] in **Section 20.3.4, RIICnMR1 — I2C Bus Mode Register 1**.

**Note 2.** The SCL line rising time [tr] and SCL line falling time [tf] depend on the total bus line capacitance [Cb] and the pull-up resistor [Rp]. For details, see the I2C bus standard from NXP Semiconductors.

**Table 20.24** lists examples of the RIICnBRH and RIICnBRL register settings when the SCL synchronization circuit is not used.

**Table 20.24 Examples of RIICnBRH/RIICnBRLSettings for Transfer Rate**

| Transfer Rate (kbps) | PCLK Frequency (MHz) | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 8 | | | | | 10 | | | | | 12.5 | | | | |
| | | RIICnBRH | | RIICnBRL | | | RIICnBRH | | RIICnBRL | | | RIICnBRH | | RIICnBRL | |
| | CKS [2:0] | SCL Clock Width (high level) [IICφ] | Setting Value [write value] | SCL Clock Width (low level) [IICφ] | Setting Value [write value] | CKS [2:0] | SCL Clock Width (high level) [IICφ] | Setting Value [write value] | SCL Clock Width (low level) [IICφ] | Setting Value [write value] | CKS [2:0] | SCL Clock Width (high level) [IICφ] | Setting Value [write value] | SCL Clock Width (low level) [IICφ] | Setting Value [write value] |
| 10 | $100_B$ | 22 | $F6_H$ | 25 | $F9_H$ | $101_B$ | 13 | $ED_H$ | 15 | $EF_H$ | $101_B$ | 16 | $F0_H$ | 20 | $F4_H$ |
| 50 | $010_B$ | 16 | $F0_H$ | 19 | $F3_H$ | $010_B$ | 21 | $F5_H$ | 24 | $F8_H$ | $011_B$ | 12 | $EC_H$ | 15 | $EF_H$ |
| 100 | $001_B$ | 15 | $EF_H$ | 18 | $F2_H$ | $001_B$ | 19 | $F3_H$ | 23 | $F7_H$ | $001_B$ | 24 | $F8_H$ | 29 | $FD_H$ |
| 400 | $000_B$ | 4 | $E4_H$ | 10 | $EA_H$ | $000_B$ | 5 | $E5_H$ | 12 | $EC_H$ | $000_B$ | 7 | $E7_H$ | 16 | $F0_H$ |

| Transfer Rate (kbps) | PCLK Frequency (MHz) | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 16 | | | | | 20 | | | | | 25 | | | | |
| | | RIICnBRH | | RIICnBRL | | | RIICnBRH | | RIICnBRL | | | RIICnBRH | | RIICnBRL | |
| | CKS [2:0] | SCL Clock Width (high level) [IICφ] | Setting Value [write value] | SCL Clock Width (low level) [IICφ] | Setting Value [write value] | CKS [2:0] | SCL Clock Width (high level) [IICφ] | Setting Value [write value] | SCL Clock Width (low level) [IICφ] | Setting Value [write value] | CKS [2:0] | SCL Clock Width (high level) [IICφ] | Setting Value [write value] | SCL Clock Width (low level) [IICφ] | Setting Value [write value] |
| 10 | $101_B$ | 22 | $F6_H$ | 25 | $F9_H$ | $110_B$ | 13 | $ED_H$ | 15 | $EF_H$ | $110_B$ | 16 | $F0_H$ | 20 | $F4_H$ |
| 50 | $011_B$ | 16 | $F0_H$ | 19 | $F3_H$ | $011_B$ | 21 | $F5_H$ | 24 | $F8_H$ | $100_B$ | 12 | $EC_H$ | 15 | $EF_H$ |
| 100 | $010_B$ | 15 | $EF_H$ | 18 | $F2_H$ | $010_B$ | 19 | $F3_H$ | 23 | $F7_H$ | $010_B$ | 24 | $F8_H$ | 29 | $FD_H$ |
| 400 | $000_B$ | 9 | $E9_H$ | 20 | $F4_H$ | $000_B$ | 11 | $EB_H$ | 25 | $F9_H$ | $001_B$ | 7 | $E7_H$ | 16 | $F0_H$ |

| Transfer Rate (kbps) | PCLK Frequency (MHz) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 30 | | | | | 33 | | | | |
| | | RIICnBRH | | RIICnBRL | | | RIICnBRH | | RIICnBRL | |
| | CKS [2:0] | SCL Clock Width (high level) [IICφ] | Setting Value [write value] | SCL Clock Width (low level) [IICφ] | Setting Value [write value] | CKS [2:0] | SCL Clock Width (high level) [IICφ] | Setting Value [write value] | SCL Clock Width (low level) [IICφ] | Setting Value [write value] |
| 10 | $110_B$ | 20 | $F4_H$ | 24 | $F8_H$ | $110_B$ | 22 | $F6_H$ | 26 | $FA_H$ |
| 50 | $100_B$ | 15 | $EF_H$ | 18 | $F2_H$ | $100_B$ | 17 | $F1_H$ | 20 | $F4_H$ |
| 100 | $011_B$ | 14 | $EE_H$ | 17 | $F1_H$ | $011_B$ | 16 | $F0_H$ | 19 | $F3_H$ |
| 400 | $001_B$ | 8 | $E8_H$ | 19 | $F3_H$ | $001_B$ | 9 | $E9_H$ | 21 | $F5_H$ |

**CAUTION**

CBRH/ICBRL settings in these tables are calculated using the following values:
SCL line rising time (tr): 100 kbps or less, [Sm]: 1000 ns, 400 kbps or less, [Fm]: 300 ns
SCL line falling time (tf): 400 kbps or less, [Sm/Fm]: 300 ns
For the specified values of SCL line rising time (tr) and SCL line falling time (tf), see the I2C bus standard from NXP Semiconductors.

## 20.3.15 RIICnDRT — I2C Bus Transmit Data Register

**Access:** RIICnDRT register can be read or written in 32-bit units.
RIICnDRTL register can be read or written in 16-bit units.
RIICnDRTLL register can be read or written in 8-bit units.

**Address:** RIICnDRT: <RIICn_base> + $003C_H$

RIICnDRTL: <RIICn_base> + $003C_H$

RIICnDRTLL: <RIICn_base> + $003C_H$

**Value after reset:** $0000\ 00FF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | DRT[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

When RIICnDRT detects a space in the I2C bus shift register (RIICnDRS), it transfers the transmit data that has been written to RIICnDRT to RIICnDRS and starts transmitting data in transmit mode.

The double-buffer structure of RIICnDRT and RIICnDRS allows continuous transmit operation if the next transmit data has been written to RIICnDRT while the RIICnDRS data is being transmitted.

RIICnDRT can always be read and written. Write transmit data to RIICnDRT once when a transmit data empty interrupt (INTRIICnTI) request is generated. When writing to bit 31 to 8, write the value after reset.

## 20.3.16 RIICnDRR — I2C Bus Receive Data Register

**Access:** RIICnDRR register is a read-only register that can be read in 32-bit units.
RIICnDRRL register is a read-only registers that can be read in 16-bit units.
RIICnDRRLL register is a read-only registers that can be read in 8-bit units.

**Address:** RIICnDRR: <RIICn_base> + $0040_H$

RIICnDRRL: <RIICn_base> + $0040_H$

RIICnDRRLL: <RIICn_base> + $0040_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | DRR[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

When 1 byte of data has been received, the received data is transferred from the I2C bus shift register (RIICnDRS) to RIICnDRR to enable the next data to be received.

The double-buffer structure of RIICnDRS and RIICnDRR allows continuous receive operation if the received data has been read from RIICnDRR while RIICnDRS is receiving data.

RIICnDRR cannot be written. Read data from RIICnDRR once when a receive complete interrupt (INTRIICnRI) request is generated.

If RIIC receives the next receive data before the current data is read from RIICnDRR (while the RIICnSR2.RDRF flag is 1), the RIIC automatically holds the SCL clock low one cycle before the RDRF flag is set to 1 next.

## 20.3.17 RIICnDRS — I2C Bus Shift Register

**Access:** This register is not accessible.

**Address:** —

**Value after reset:** 0000 00FF$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | DRS[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |

RIICnDRS is a shift register to transmit and receive data.

During transmission, transmit data is transferred from RIICnDRT to RIICnDRS and is sent from the SDA pin. During reception, data is transferred from RIICnDRS to RIICnDRR after 1 byte of data has been received.

RIICnDRS cannot be accessed directly.

## 20.4 Interrupt Sources

The RIIC issues four types of interrupt request: transfer error or event generation (arbitration-lost detection, NACK detection, timeout detection, start condition detection, and stop condition detection), receive complete, transmit data empty, and transmit end.

**Table 20.25** lists details of the several interrupt requests. The receive complete and transmit data empty interrupt request are both capable of launching data transfer by the DMAC.

**Table 20.25 Interrupt Sources**

| Symbol | Interrupt Source | Interrupt Flag | DMACA Launching | Interrupt Condition |
|---|---|---|---|---|
| INTRIICnTI | Transmit Data Empty | TDRE | Possible | TDRE = 1 and TIE = 1 |
| INTRIICnTEI | Transmit End | TEND | Not possible | TEND = 1 and TEIE = 1 |
| INTRIICnRI | Receive Complete | RDRF | Possible | RDRF = 1 and RIE = 1 |
| INTRIICnEE | Transfer Error/ Event Generation | AL | Not possible | AL = 1 and ALIE = 1 |
| | | NACKF | | NACKF = 1 and NAKIE = 1 |
| | | TMOF | | TMOF = 1 and TMOIE = 1 |
| | | START | | START = 1 and STIE = 1 |
| | | STOP | | STOP = 1 and SPIE = 1 |

Clear or mask the each flag during interrupt handling.

**CAUTIONS**

1. **There is a latency (delay) between the execution of a write instruction for a peripheral module by the CPU and actual writing to the module. Thus, when an interrupt flag has been cleared or masked, read the relevant flag again to check whether clearing or masking has been completed, and then return from interrupt processing. Returning from interrupt processing without checking that writing to the module has been completed creates a possibility of repeated processing of the same interrupt.**
2. **Since INTRIICnTI is an edge-detected interrupt, it does not require clearing. Furthermore, the RIICnSR2.TDRE flag (a condition for INTRIICnTI) is automatically cleared to 0 when data for transmission are written to RIICnDRT or a stop condition is detected (RIICnSR2.STOP flag = 1).**
3. **Since INTRIICnRI is an edge-detected interrupt, it does not require clearing. Furthermore, the RIICnSR2.RDRF flag (a condition for INTRIICnRI) is automatically cleared to 0 when data are read from RIICnDRR.**
4. **When using the INTRIICnTEI interrupt, clear the RIICnSR2.TEND flag in the INTRIICnTEI interrupt processing.**
   **Note that the RIICnSR2.TEND flag is automatically cleared to 0 when data for transmission are written to RIICnDRT or a stop condition is detected (RIICnSR2.STOP flag = 1).**

# 20.5 Operation

## 20.5.1 Communication Data Format

The I2C bus format consists of 8-bit data and 1-bit acknowledge. The frame following a start condition or restart condition is an address frame used to specify a slave device with which the master device communicates. The specified slave is valid until a new slave is specified or a stop condition is issued.

**Figure 20.3** shows the I2C bus format, and **Figure 20.4** shows the I2C bus timing.

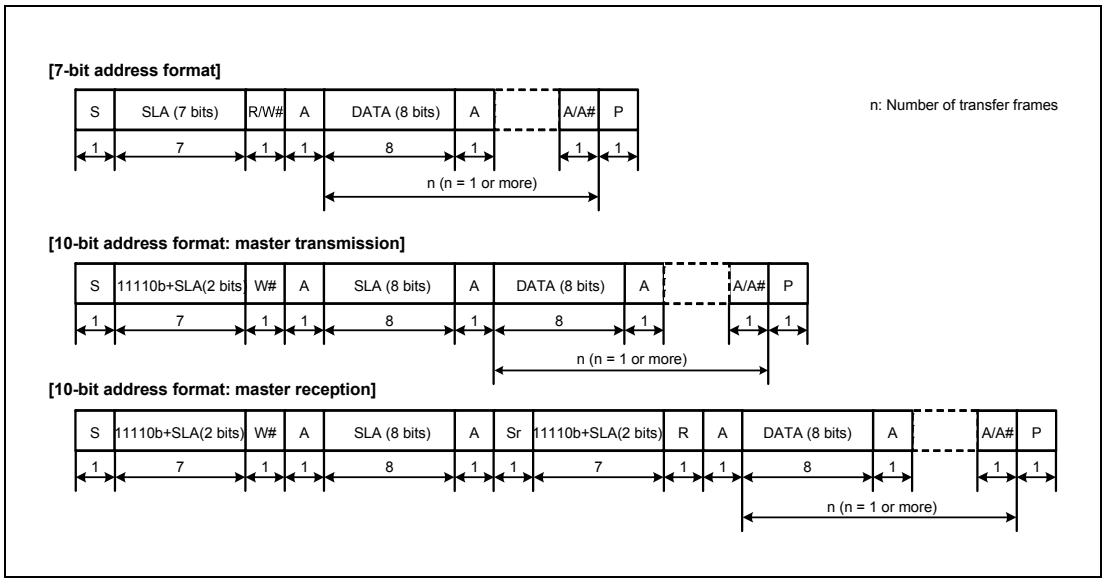



Figure 20.3 I2C Bus Format

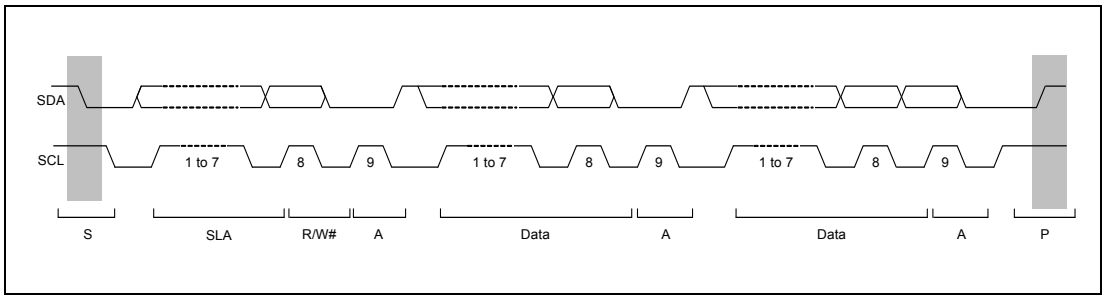



Figure 20.4 I2C Bus Timing (SLA = 7 Bits)

S: Start condition. The master device drives the SDA line low from high level while the SCL line is at a high level.

SLA: Slave address, by which the master device selects a slave device.

R/W#: Indicates the direction of data transfer: from the slave device to the master device when R/W is 1, or from the master device to the slave device when R/W is 0.

A: Acknowledge. The receive device drives the SDA line low. (In master transmit mode, the slave device returns acknowledge. In master receive mode, the master device returns acknowledge.)

A#: Not-acknowledge. The receiving device has not returned a response or is not present so the SDA line has remained at the high level.

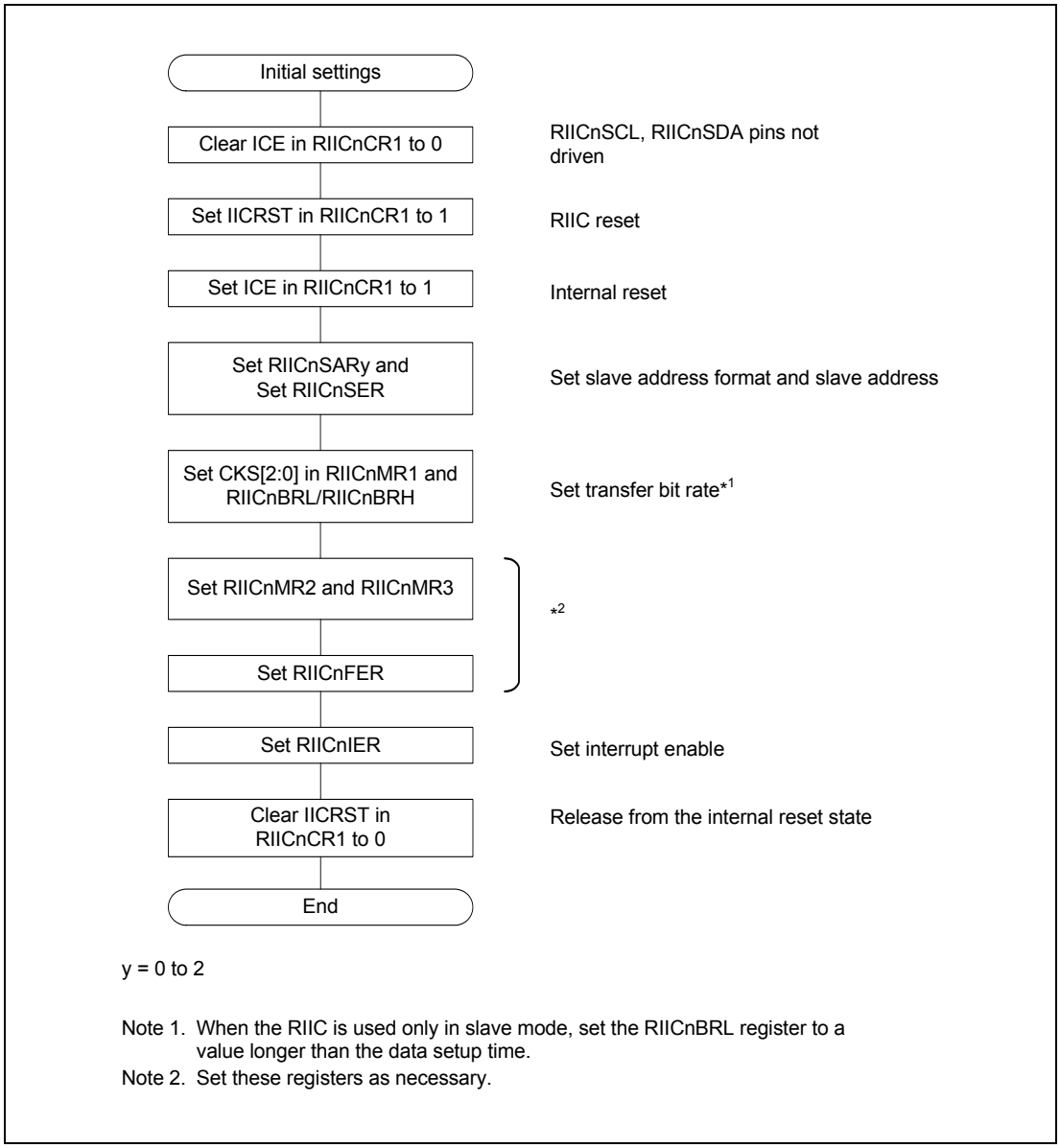
Sr: Restart condition. The master device drives the SDA line low from the high level after the setup time has elapsed with the SCL line at the high level.

DATA: Transmitted or received data

P: Stop condition. The master device drives the SDA line high from low level while the SCL line is at a high level.

## 20.5.2 Initial Settings

Before starting data transmission and reception, initialize the RIIC according to the procedure in **Figure 20.5**. Make initial settings for the RIIC once when starting the RIIC.




Note 1. When the RIIC is used only in slave mode, set the RIICnBRL register to a value longer than the data setup time.

Note 2. Set these registers as necessary.

**Figure 20.5 Example of RIIC Initialization Flowchart**

### 20.5.3 Master Transmit Operation

In master transmit operation, the RIIC outputs the SCL (clock) and transmitted data signals as the master device, and the slave device returns acknowledgements. **Figure 20.6** shows an example of usage of master transmission and **Figure 20.7** to **Figure 20.9** show the timing of operations in master transmission.

The following describes the procedure and operations for master transmission.

(1) Set the RIICnCR1.IICRST bit 1 to 1 (RIIC reset) and then set the RIICnCR1.ICE bit to 1 (internal reset) with the RIICnCR1.ICE bit cleared to 0 (RIICnSCL and RIICnSDA pins not driven). This initializes the internal state and the various flags of RIICnSR1. After that, set registers RIICnSARy, RIICnSER, RIICnMR1, RIICnBRH, and RIICnBRL (y = 0 to 2), and set the other registers as necessary (for initial settings of the RIIC, see **Figure 20.5**). When the necessary register settings have been completed, set the RIICnCR1.IICRST bit to 0 (for release from the reset state). This step is not necessary if initialization of the RIIC has already been completed.

(2) Read the RIICnCR2.BBSY flag to check that the bus is open, and then set the RIICnCR2.ST bit to 1 (start condition issuance request). Upon receiving the request, the RIIC issues a start condition. At the same time, the BBSY flag and the RIICnSR2.START flag are automatically set to 1 and the ST bit is automatically cleared to 0. At this time, if the start condition is detected and the internal levels for the SDA output state and the levels on the SDA line have matched while the ST bit is 1, the RIIC recognizes that issuing of the start condition as requested by the ST bit has been successfully completed, and the RIICnCR2.MST and TRS bits are automatically set to 1, placing the RIIC in master transmit mode. The RIICnSR2.TDRE flag is also automatically set to 1 in response to setting of the TRS and MST bits to 1.

(3) Check that the RIICnSR2.TDRE flag is 1, and then write the value for transmission (the slave address and the R/W# bit) to RIICnDRT. Once the data for transmission are written to RIICnDRT, the TDRE flag is automatically cleared to 0, the data are transferred from RIICnDRT to RIICnDRS, and the TDRE flag is again set to 1. After the slave address including the R/W# bit has been transmitted, the value of the TRS bit is automatically updated to select master transmit or master receive mode in accord with the value of the transmitted R/W# bit. If the value of the R/W# bit was 0, the RIIC continues in master transmit mode.
Since the RIICnSR2.NACKF flag being 1 at this time indicates that the slave address has not been recognized or there was an error in communications, write 1 to the RIICnCR2.SP bit to issue a stop condition.
For data transmission with an address in the 10-bit format, start by writing 1111 $0_B$, the two higher-order bits of the slave address, and W# to RIICnDRT as the first address transmission. Then, as the second address transmission, write the eight lower-order bits of the slave address to RIICnDRT.

(4) After confirming that the RIICnSR2.TDRE flag is 1, write the data for transmission to the RIICnDRT register. The RIIC automatically holds the SCL line low until the data for transmission are ready or a stop condition is issued.

(5) After all bytes of data for transmission have been written to the RIICnDRT register, wait until the value of the RIICnSR2.TEND flag returns to 1, and then set the RIICnCR2.SP bit to 1 (stop condition issuance request). Upon receiving a stop condition issuance request, the RIIC issues the stop condition.

(6) Upon detecting the stop condition, the RIIC automatically clears the RIICnCR2.MST and TRS bits to $00_B$ and enters slave receive mode. Furthermore, it automatically clears the RIICnSR2.TDRE and TEND flags to 0, and sets the RIICnSR2.STOP flag in to 1.

(7) After checking that the RIICnSR2.STOP flag is 1, clear the RIICnSR2.NACKF and STOP flags to 0 for the next transfer operation.

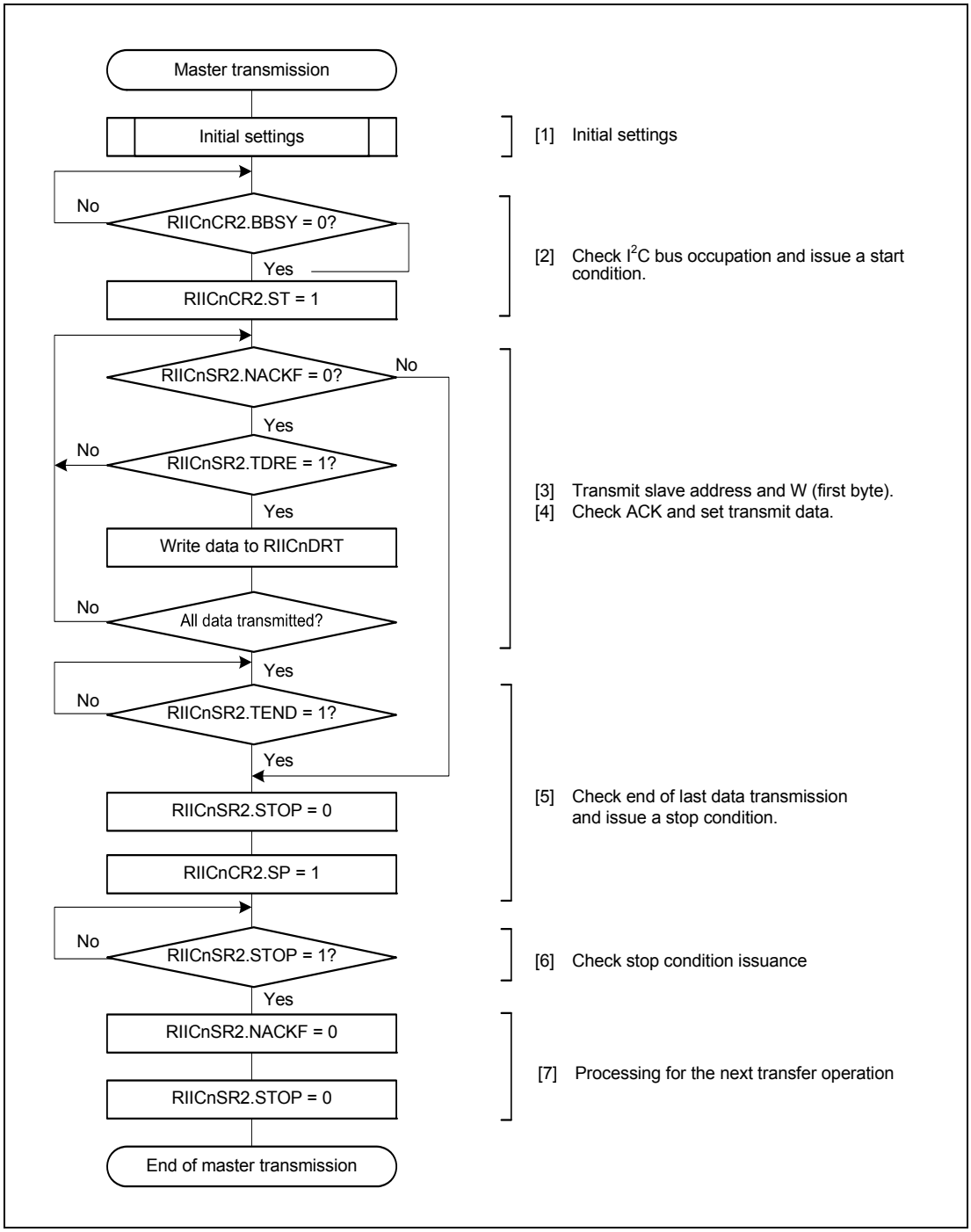



**Figure 20.6 Example of Master Transmission Flowchart**

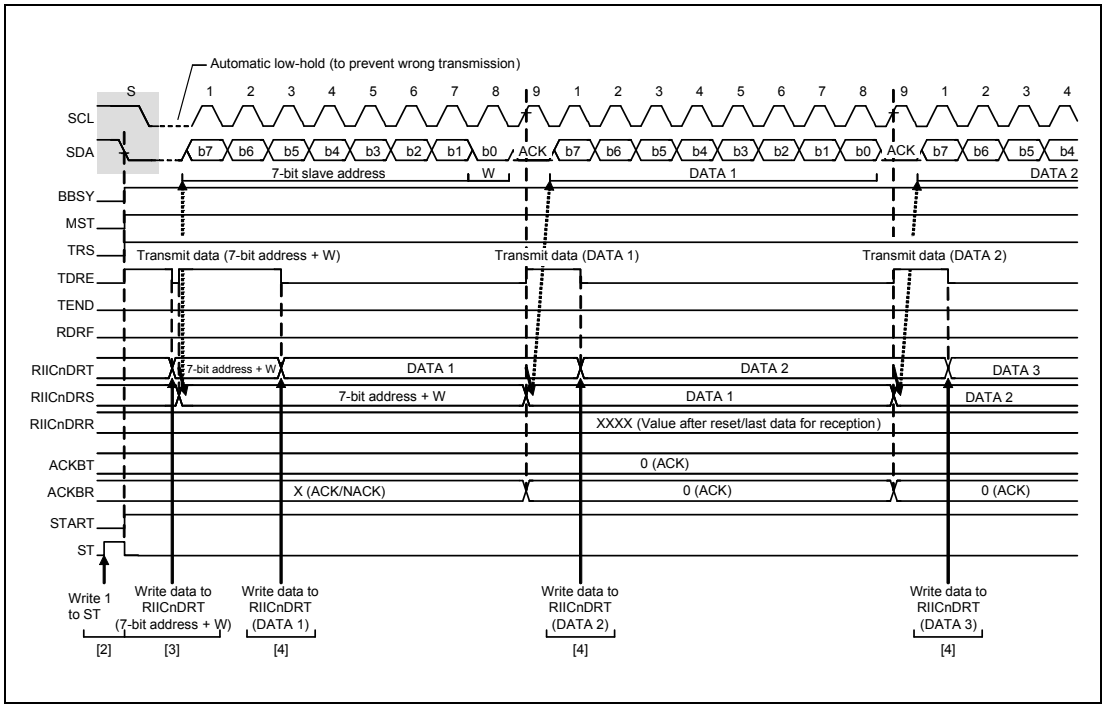



**Figure 20.7 Master Transmit Operation Timing (1) (7-Bit Address Format)**

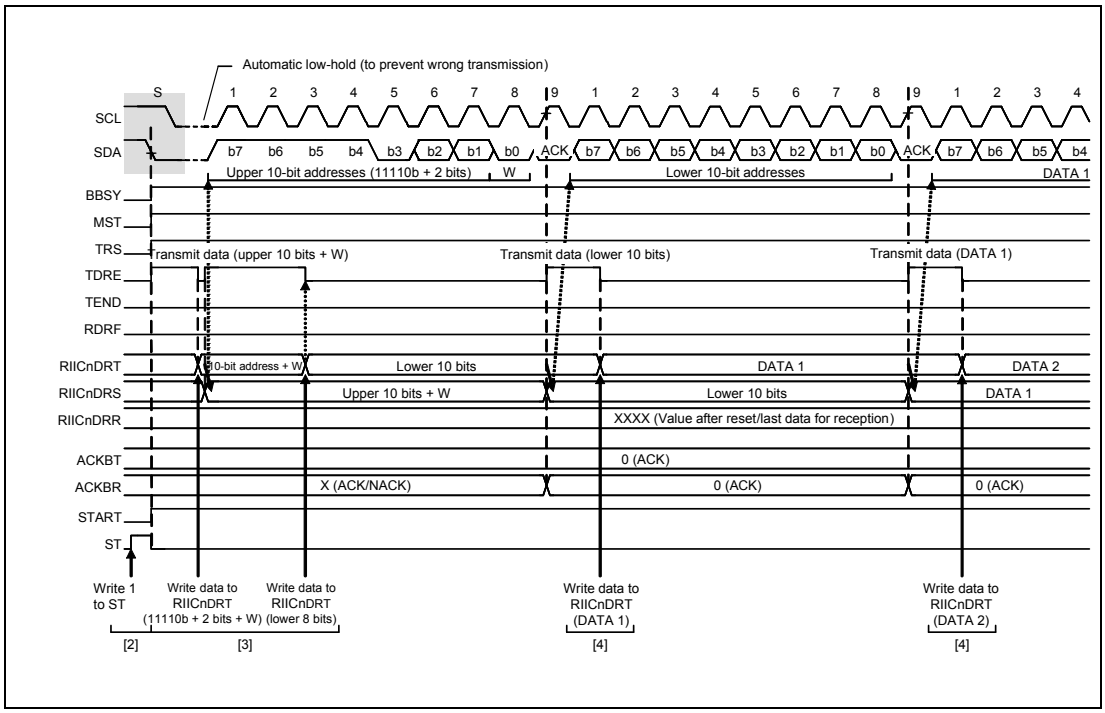



**Figure 20.8 Master Transmit Operation Timing (2) (10-Bit Address Format)**

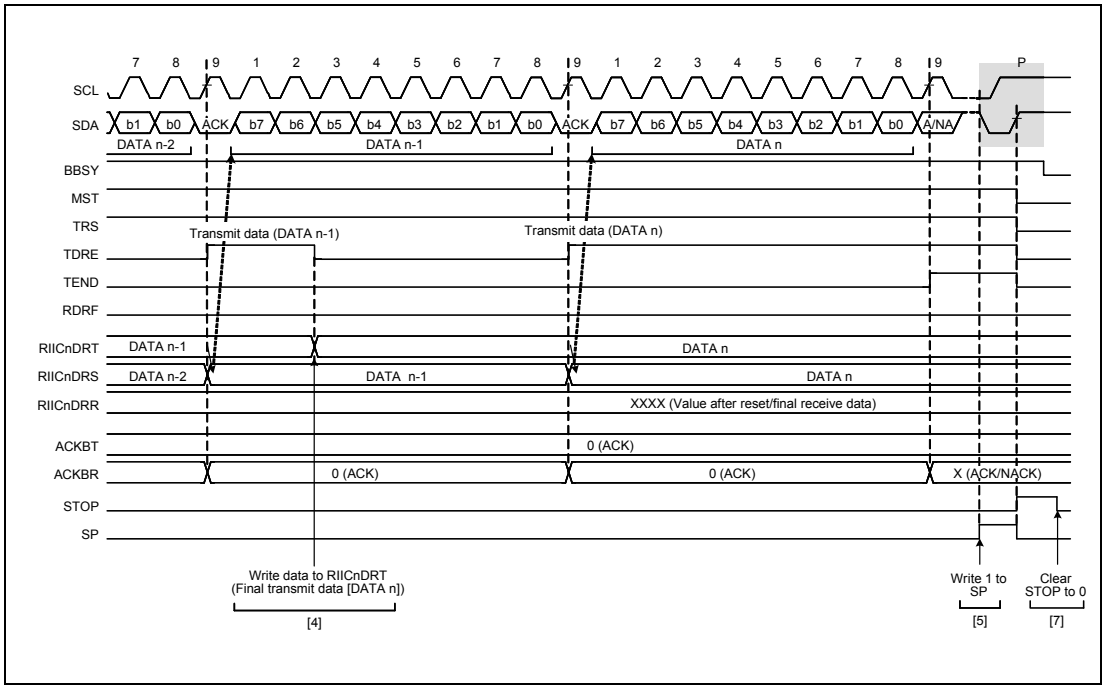



**Figure 20.9 Master Transmit Operation Timing (3)**

### 20.5.4 Master Receive Operation

In master receive operation, the RIIC as a master device outputs the SCL (clock) signal, receives data from the slave device, and returns acknowledgements. Since the RIIC must start by sending a slave address to the corresponding slave device, this part of the procedure is performed in master transmit mode, but the subsequent steps are in master receive mode.

**Figure 20.10** shows an example of master reception flowchart (7-bit address format, 1 or 2 bytes), **Figure 20.11** shows an example of master reception flowchart (7-bit address format, 3 bytes or more), and **Figure 20.12** to **Figure 20.14** show the timing of operations in master reception.

The following describes the procedure and operations for master reception.

(1) Set the RIICnCR1.IICRST bit to 1 (RIIC reset) and then set the RIICnCR1.ICE bit to 1 (internal reset) with the RIICnCR1.ICE bit cleared to 0 (RIICnSCL and RIICnSDA pins not driven). This initializes the internal state and the various flags of RIICnSR1. After that, set registers RIICnSARy, RIICnSER, RIICnMR1, RIICnBRH, and RIICnBRL (y = 0 to 2), and set the other registers as necessary (for initial settings of the RIIC, see **Figure 20.5**). When the necessary register settings have been completed, set the RIICnCR1.IICRST bit to 0 (for release from the reset state). This step is not necessary if initialization of the RIIC has already been completed.

(2) Read the RIICnCR2.BBSY flag to check that the bus is open, and then set the RIICnCR2.ST bit to 1 (start condition issuance request). Upon receiving the request, the RIIC issues a start condition. When the RIIC detects the start condition, the BBSY flag and the RIICnSR2.START flag are automatically set to 1 and the ST bit is automatically cleared to 0. At this time, if the start condition is detected and the levels for the SDA output and the levels on the SDA line have matched while the ST bit is 1, the RIIC recognizes that issuing of the start condition as requested by the ST bit has been successfully completed, and the RIICnCR2.MST and TRS bits are automatically set to 1, placing the RIIC in master transmit mode. The RIICnSR2.TDRE flag is also automatically set to 1 in response to setting of the TRS bit to 1.

(3) Check that the RIICnSR2.TDRE flag is 1, and then write the value for transmission (the slave address and value of the R/W# bit) to RIICnDRT. Once the data for transmission are written to RIICnDRT, the TDRE flag is automatically cleared to 0, the data are transferred from RIICnDRT to RIICnDRS, and the TDRE flag is again set to 1. After the slave address including the R/W# bit has been transmitted, the value of the RIICnCR2.TRS bit is automatically updated to select transmit or receive mode in accord with the value of the transmitted R/W# bit. If the value of the R/W# bit was 1, the RIICnCR2.TRS bit is cleared to 0 on the rising edge of the ninth cycle of SCL (the clock signal), placing the RIIC in master receive mode. At this time, the TDRE flag is automatically cleared to 0 and the RIICnSR2.RDRF flag is automatically set to 1.
Since the RIICnSR2.NACKF flag being 1 at this time indicates that the slave address has not been recognized or there was an error in communications, write 1 to the RIICnCR2.SP bit to issue a stop condition.
For master reception from a device with a 10-bit address, start by using master transmission to transmit the two higher-order bits of the slave address and then the eight lower-order bits of the slave address, and issue a restart condition following generation of the transmission end interrupt (or after TEND = 1) (see **Figure 20.13** for operation timing). After that, transmitting 1111 $0_B$ plus the two higher-order bits of the slave address, and the R bit places the RIIC in master receive mode.

(4) Dummy read RIICnDRR after confirming that the RIICnSR2.RDRF flag is 1; this makes the RIIC start output of the SCL (clock) signal and start data reception.

(5) After 1 byte of data has been received, the RIICnSR2.RDRF flag is set to 1 on the rising edge of the eighth or ninth cycle of SCL clock (the clock signal) as selected by the RIICnMR3.RDRFS bit. Reading out RIICnDRR at this time will produce the received data, and the RDRF flag is automatically cleared to 0 at the same time. Furthermore, the value of the acknowledgement field received during the ninth cycle of SCL clock is returned as the value set in the RIICnMR3.ACKBT bit. Furthermore, if the next byte to be received is the next to last byte, set the RIICnMR3.WAIT bit to 1 (for wait insertion) before reading the RIICnDRR (containing the second byte from last). As well as enabling NACK output even in the case of delays in processing to set the RIICnMR3.ACKBT bit to 1 (NACK) in step (6), due to other interrupts, etc., this fixes the SCL line to the low level on the rising edge of the ninth clock cycle in reception of the last byte, so the state is such that issuing a stop condition is possible.

(6) When the RIICnMR3.RDRFS bit is 0 and the slave device must be notified that it is to end transfer for data reception after transfer of the next (final) byte, set the RIICnMR3.ACKBT bit to 1 (NACK).

(7) After reading out the byte before last from the RIICnDRR register, if the value of the RIICnSR2.RDRF flag is confirmed to be 1, write 1 to the RIICnCR2.SP bit (stop condition issuance request) and then read the last byte from RIICnDRR. When RIICnDRR is read, the RIIC is released from the wait state and issues the stop condition after low-level output in the ninth clock cycle is completed or the SCL line is released from the low-hold state.

(8) Upon detecting the stop condition, the RIIC automatically clears the RIICnCR2.MST and TRS bits to $00_B$ and enters slave receive mode. Furthermore, detection of the stop condition leads to setting of the RIICnSR2.STOP flag to 1.

(9) After checking that the RIICnSR2.STOP flag is 1, clear the RIICnSR2.NACKF and STOP flags to 0 for the next transfer operation.

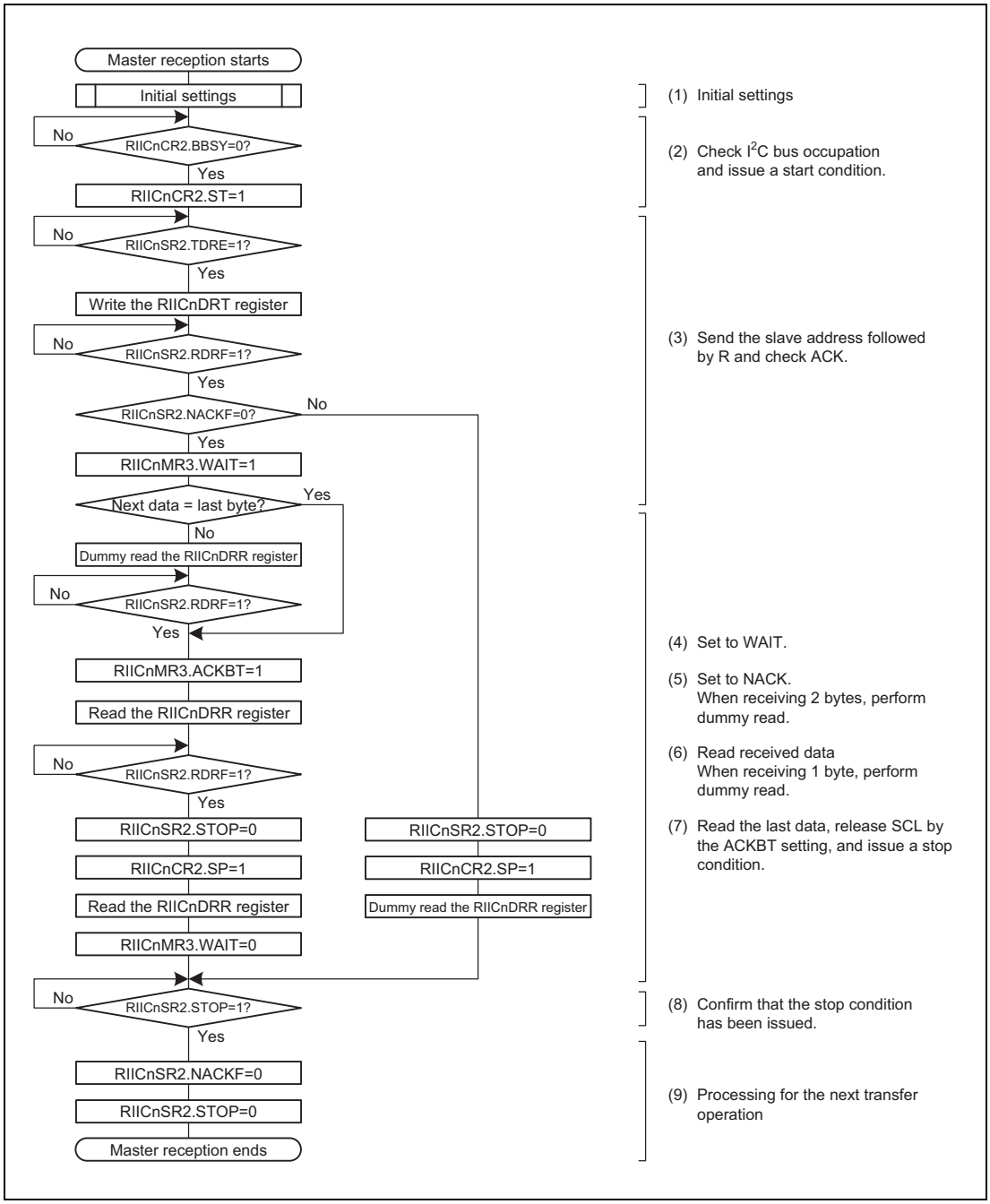



**Figure 20.10 Example of Master Reception Flowchart (7-Bit Address Format, 1 or 2 Bytes)**

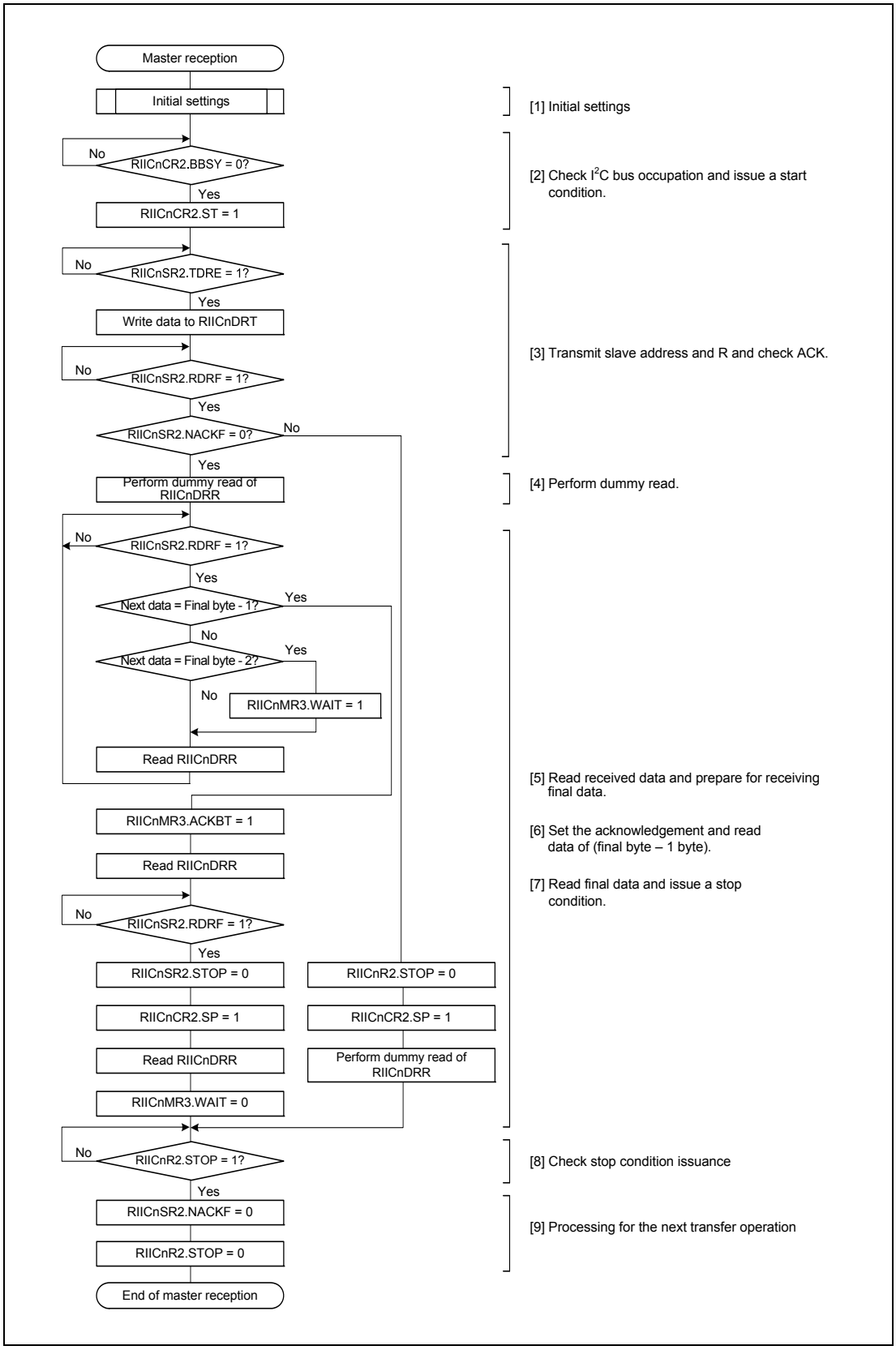



**Figure 20.11 Example of Master Reception Flowchart (7-Bit Address Format, 3 Bytes or More)**

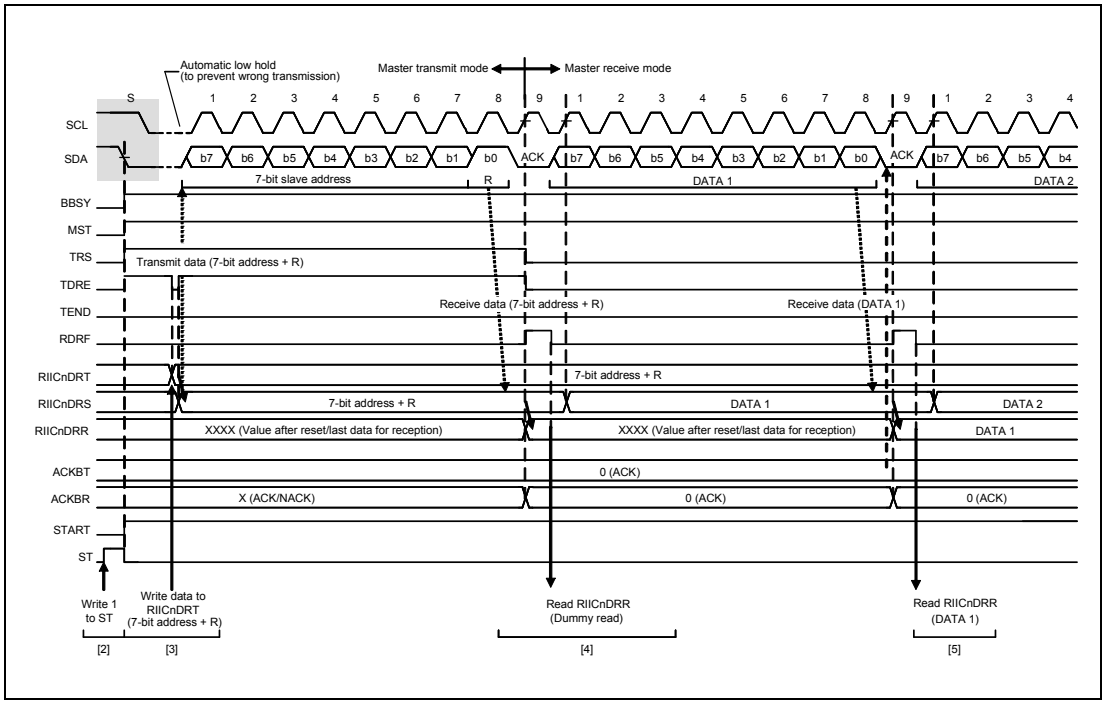



**Figure 20.12 Master Receive Operation Timing (1) (7-Bit Address Format, when RDRFS = 0)**

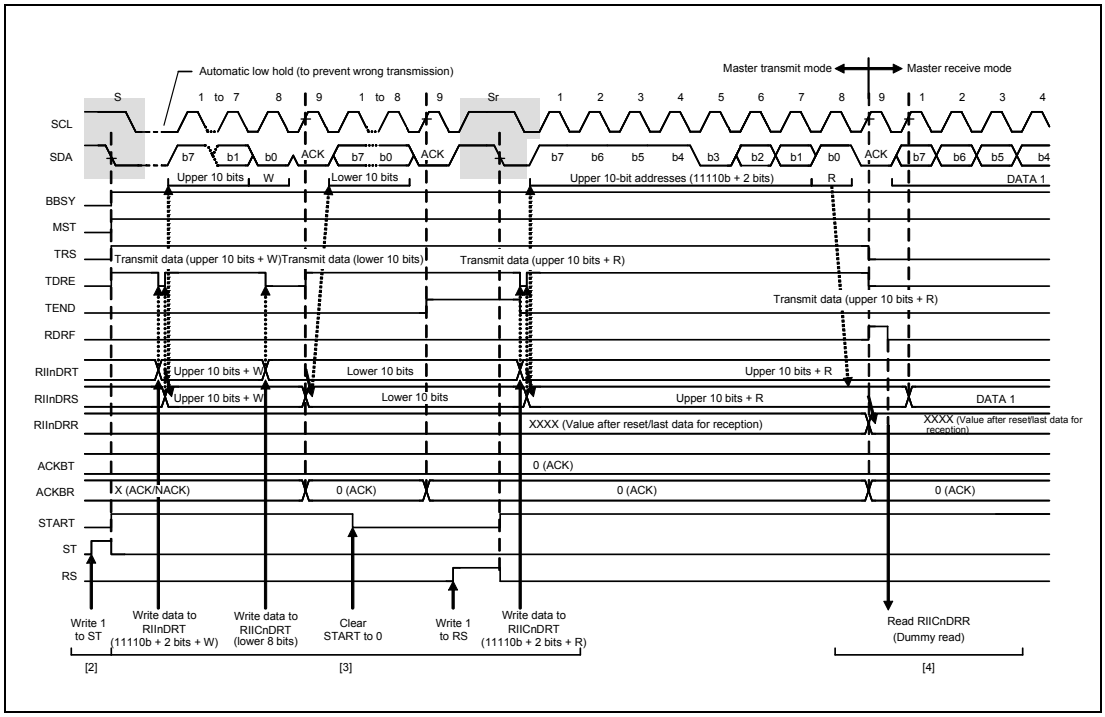



**Figure 20.13 Master Receive Operation Timing (2) (10-Bit Address Format, when RDRFS = 0)**

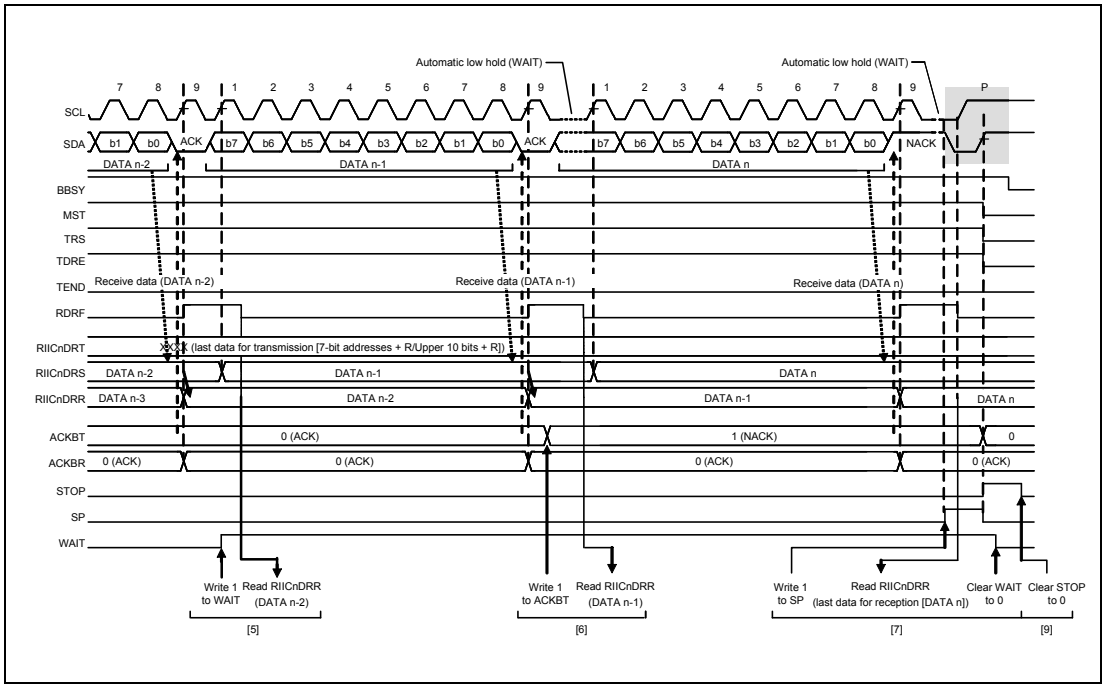



**Figure 20.14 Master Receive Operation Timing (3) (when RDRFS = 0)**

### 20.5.5 Slave Transmit Operation

In slave transmit operation, the master device outputs the SCL (clock) signal, the RIIC transmits data as a slave device, and the master device returns acknowledgements.

**Figure 20.5** shows an example of usage of slave transmission and **Figure 20.16** and **Figure 20.17** show the timing of operations in slave transmission.

The following describes the procedure and operations for slave transmission.

(1) Follow the procedure in **Figure 20.5** to make initial settings for the RIIC. This step is not necessary if initialization of the RIIC has already been completed. After initial settings, the RIIC automatically sets the BBSY and RIICnSR2.START flags to 1 and automatically clears the ST bit to 0 on detection of a start condition.

(2) After receiving a matching slave address, the RIIC sets one of the corresponding bits RIICnSR1.GCA, and AASy (y = 0 to 2) to 1 on the rising edge of the ninth cycle of SCL clock (the clock signal) and returns the value set in the RIICnMR3.ACKBT bit to the acknowledge bit on the ninth cycle of SCL clock. If the value of the R/W# bit that was also received at this time is 1, the RIIC automatically places itself in slave transmit mode by setting both the RIICnCR2.TRS bit and the RIICnSR2.TDRE flag to 1.

(3) After the RIICnSR2.TDRE flag is confirmed to be 1, write the data for transmission to the RIICnDRT register. At this time, if the RIIC receives no acknowledge from the master device (receives an NACK signal) while the RIICnFER.NACKE bit is 1, the RIIC suspends transfer of the next data.

(4) Wait until the RIICnSR2.TEND flag is set to 1 while the RIICnSR2.TDRE flag is 1, after the RIICnSR2.NACKF flag is set to 1 or the last byte for transmission is written to the RIICnDRT register. When the RIICnSR2.NACKF flag or the TEND flag is 1, the RIIC drives the SCL line low on the ninth falling edge of SCL clock.

(5) When the RIICnSR2.NACKF flag or the RIICnSR2.TEND flag is 1, dummy read RIICnDRR to complete the processing. This releases the SCL line.

(6) Upon detecting the stop condition, the RIIC automatically clears bits RIICnSR1.GCA, and AASy (y = 0 to 2), flags RIICnSR2.TDRE and TEND, and the RIICnCR2.TRS bit to 0, and enters slave receive mode. Furthermore, it automatically sets the RIICnSR2.STOP flag to 1.

(7) After checking that the RIICnSR2.STOP flag is 1, clear the RIICnSR2.NACKF and STOP flags to 0 for the next transfer operation.

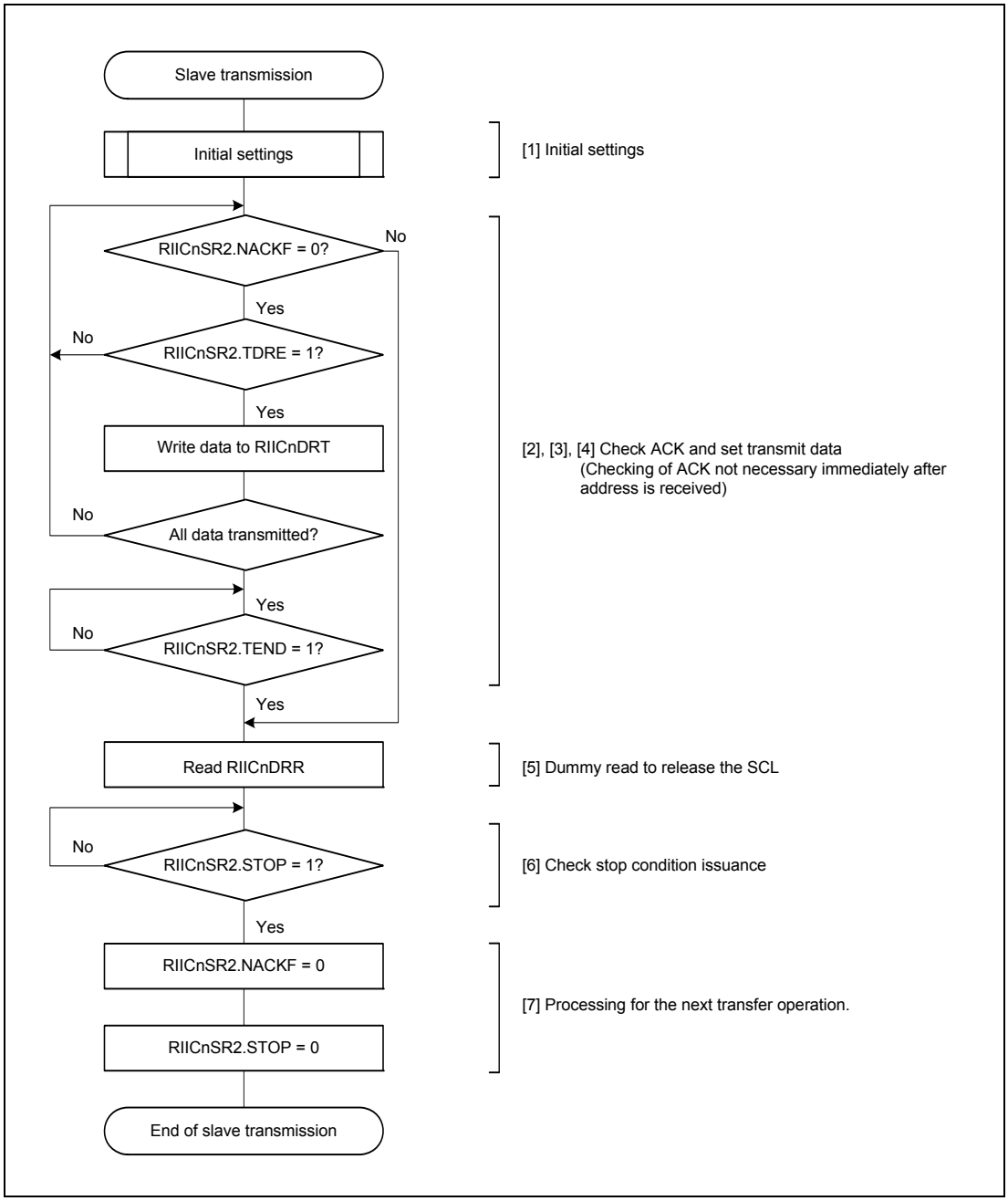



**Figure 20.15 Example of Slave Transmission Flowchart**

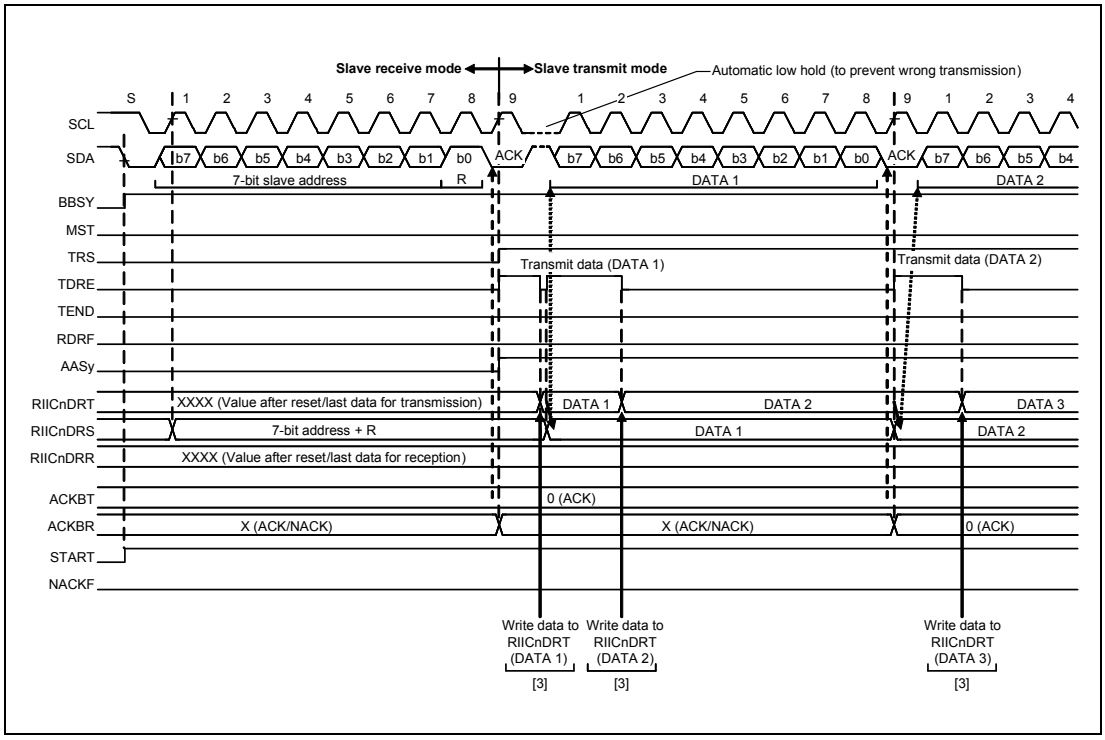



**Figure 20.16 Slave Transmit Operation Timing (1) (7-Bit Address Format)**

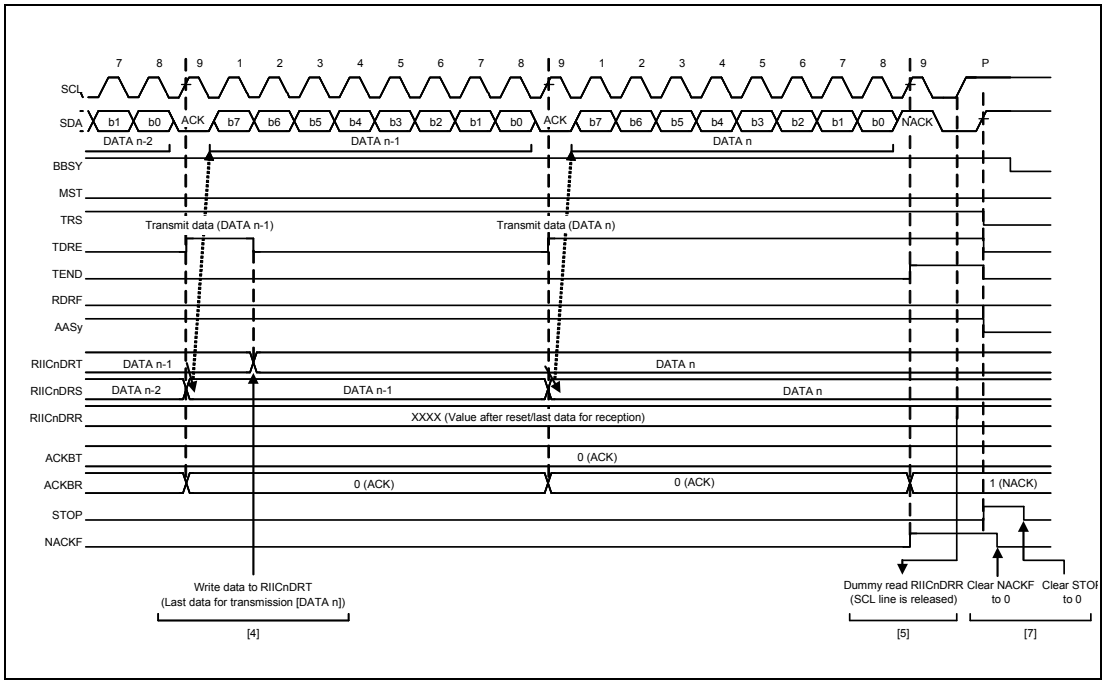



**Figure 20.17 Slave Transmit Operation Timing (2)**

### 20.5.6 Slave Receive Operation

In slave receive operation, the master device outputs the SCL clock and transmit data, and the RIIC returns acknowledgements as a slave device.

**Figure 20.18** shows an example of usage of slave reception and **Figure 20.19** and **Figure 20.20** show the timing of operations in slave reception.

The following describes the procedure and operations for slave reception.

(1) Follow the procedure in **Figure 20.5** to make initial settings for the RIIC. This step is not necessary if initialization of the RIIC has already been completed. After initial settings, the RIIC automatically sets the BBSY and RIICnSR2.START flags to 1 and automatically clears the ST bit to 0 on detection of a start condition.

(2) After receiving a matching slave address, the RIIC sets one of the corresponding bits RIICnSR1.GCA, and AASy (y = 0 to 2) to 1 on the rising edge of the ninth cycle of SCL clock (the clock signal) and returns the value set in the RIICnMR3.ACKBT bit to the acknowledge bit on the ninth cycle of SCL clock. If the value of the R/W# bit that was also received at this time is 0, the RIIC continues to place itself in slave receive mode and sets the RIICnSR2.RDRF flag to 1.

(3) After the RIICnSR2.STOP flag is confirmed to be 0 and the RIICnSR2.RDRF flag to be 1, dummy read RIICnDRR as the first read operation (the dummy value consists of the slave address and R/W# bit when the 7-bit address format is selected, or the lower eight bits when the 10-bit address format is selected).

(4) When RIICnDRR is read, the RIIC automatically clears the RIICnSR2.RDRF flag to 0. If reading of RIICnDRR is delayed and a next byte is received while the RDRF flag is still set to 1, the RIIC holds the SCL line low from one SCL cycle before the timing with which RDRF should be set. In this case, reading RIICnDRR releases the SCL line from being held at the low level.
When the RIICnSR2.STOP flag is 1 and the RIICnSR2.RDRF flag is also 1, read RIICnDRR until all the data is completely received.

(5) Upon detecting the stop condition, the RIIC automatically clears bits RIICnSR1.GCA, and AASy (y = 0 to 2) to 0. Furthermore, it automatically sets the RIICnSR2.STOP flag to 1.

(6) After checking that the RIICnSR2.STOP flag is 1, clear the RIICnSR2.STOP flag to 0 for the next transfer operation.

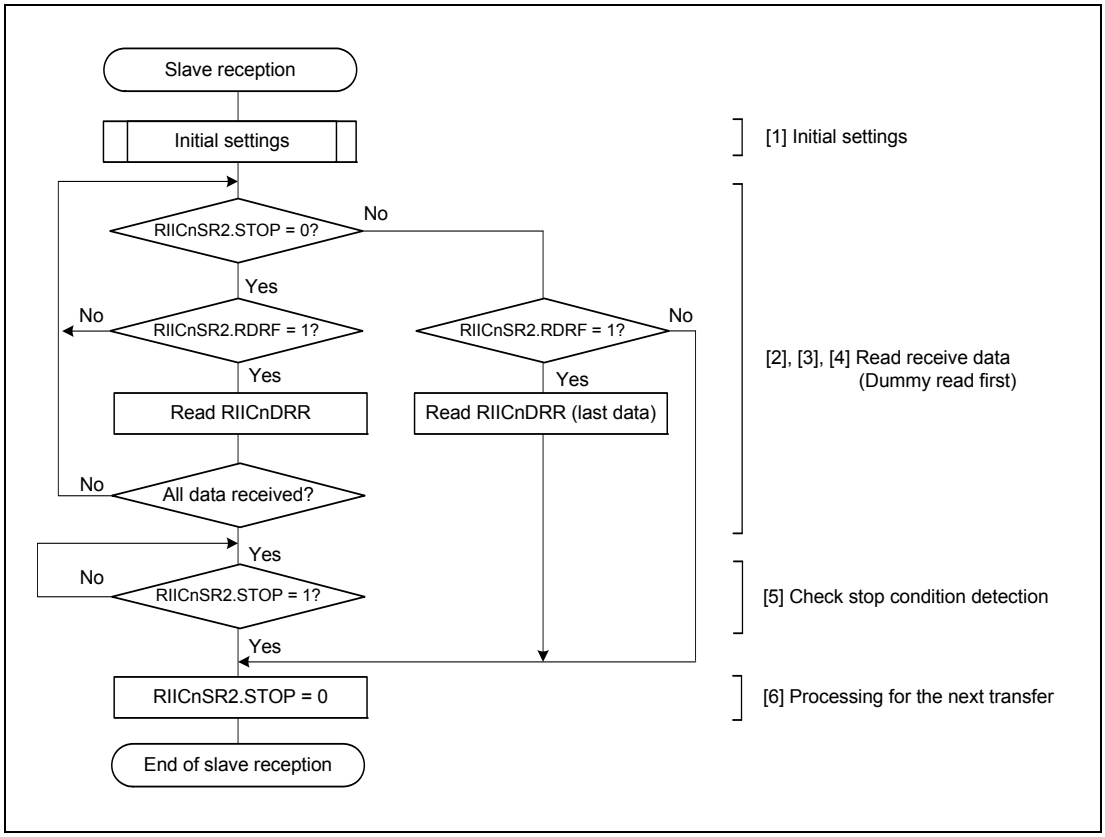



**Figure 20.18 Example of Slave Reception Flowchart**

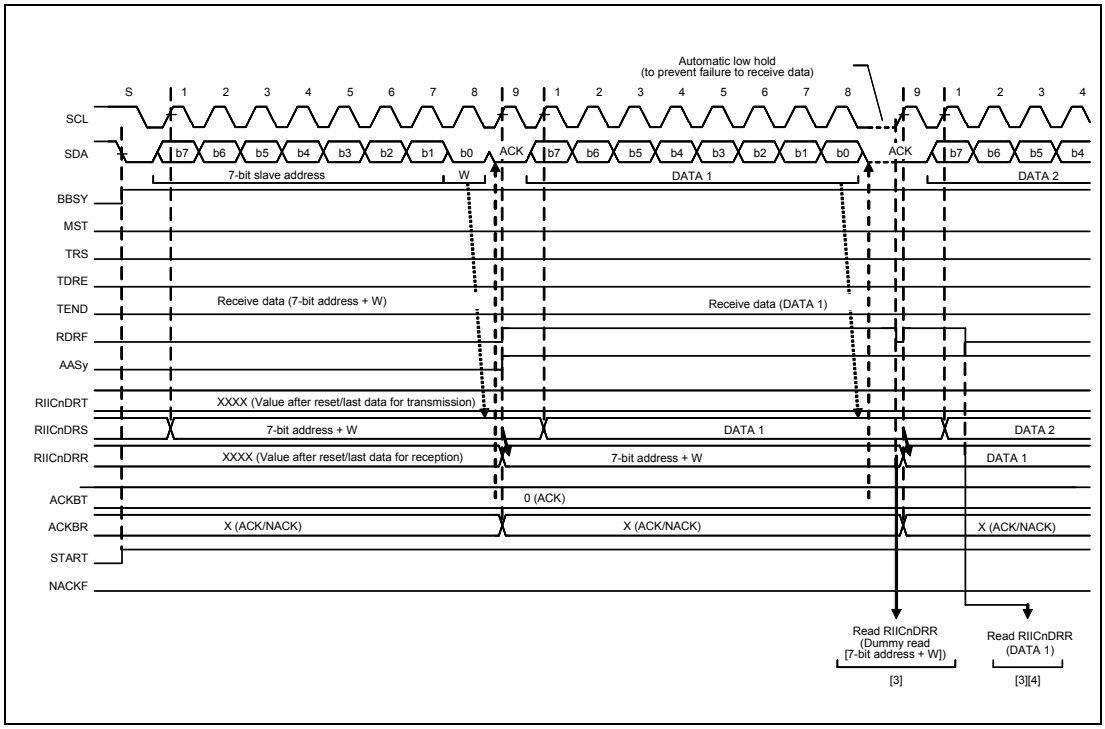



**Figure 20.19 Slave Receive Operation Timing (1) (7-Bit Address Format, when RDRFS = 0)**

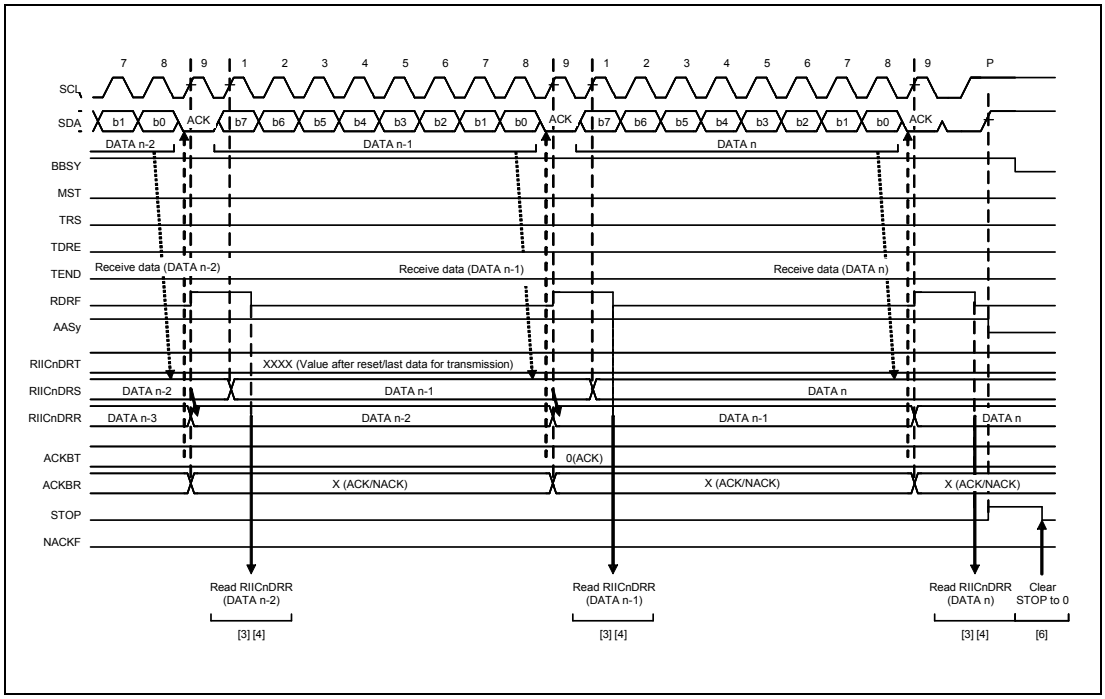



**Figure 20.20 Slave Receive Operation Timing (2) (when RDRFS = 0)**

## 20.6 SCL Synchronization Circuit

In generation of the SCL (clock) signal, the RIIC starts counting out the value for width at high level specified in RIICnBRH when it detects a rising edge on the SCL line and drives the SCL line low once counting of the width at high level is complete. When the RIIC detects the falling edge of the SCL line, it starts counting out the width at low level period specified in RIICnBRL, and then stops driving the SCL line (releases the line) once counting of the width at low level is complete. The SCL (clock) signal is thus generated.

If multiple master devices are connected to the I2C bus, a collision of SCL signals may arise due to contention with another master device. In such cases, the master devices have to synchronize their SCL signals. Since this synchronization of SCL signals must be bit by bit, the RIIC is equipped with a facility (the SCL synchronization circuit) to obtain bit-by-bit synchronization of the SCL clock signals by monitoring the SCL line during communication.

When the RIIC has detected a rising edge on the SCL line and thus started counting out the width at high level specified in RIICnBRH, and the level on the SCL line falls because an SCL signal is being generated by another master device, the RIIC stops counting when it detects the falling edge, drives the level on the SCL line low, and starts counting out the width at low level specified in RIICnBRL. When the RIIC finishes counting out the width at low level, it stops driving the SCL line to the low level (i.e. releases the line). At this time, if the width at low level of the SCL clock signal from the other master device is longer than the width at low level set in the RIIC, the width at low level of the SCL signal will be extended. Once the width at low level for the other master device has ended, the SCL signal rises because the SCL line has been released. That is, in cases of contention of SCL signals from more than one master, the width at high level of the SCL signal is synchronized with that of the clock having the narrower width, and the width at low level of the SCL signal is synchronized with that of the clock having the broader width. However, such synchronization of the SCL signal is only enabled when the RIICnFER.SCLE bit is set to 1.

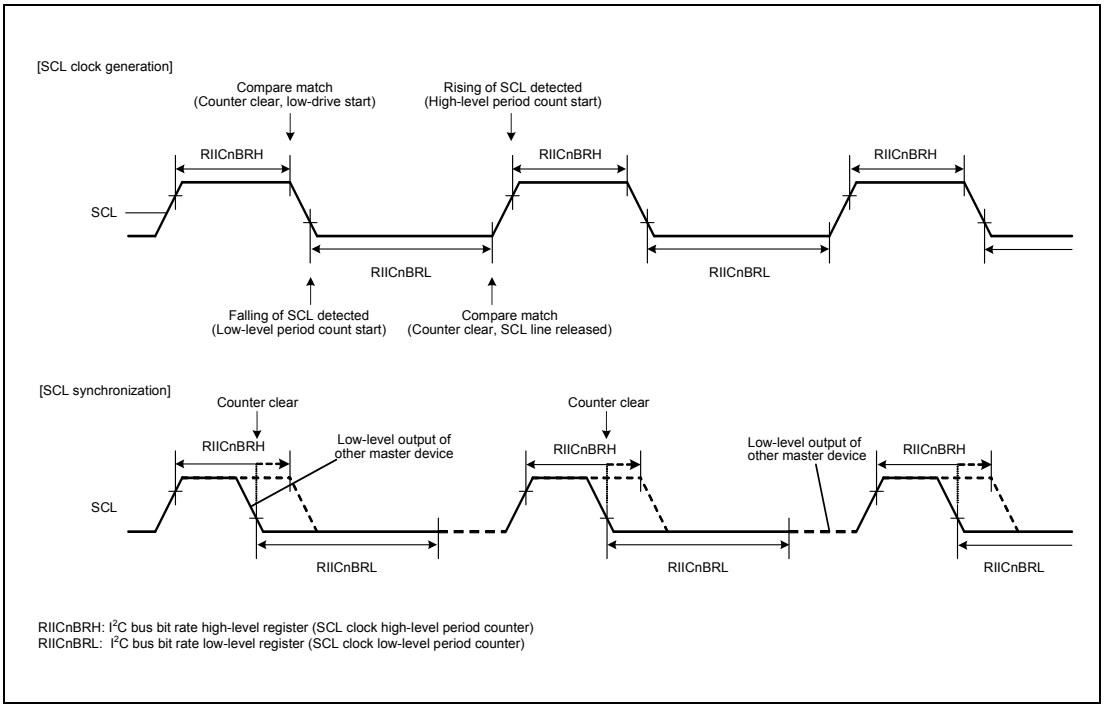



**Figure 20.21 Generation and Synchronization of the SCL Signal from the RIIC**

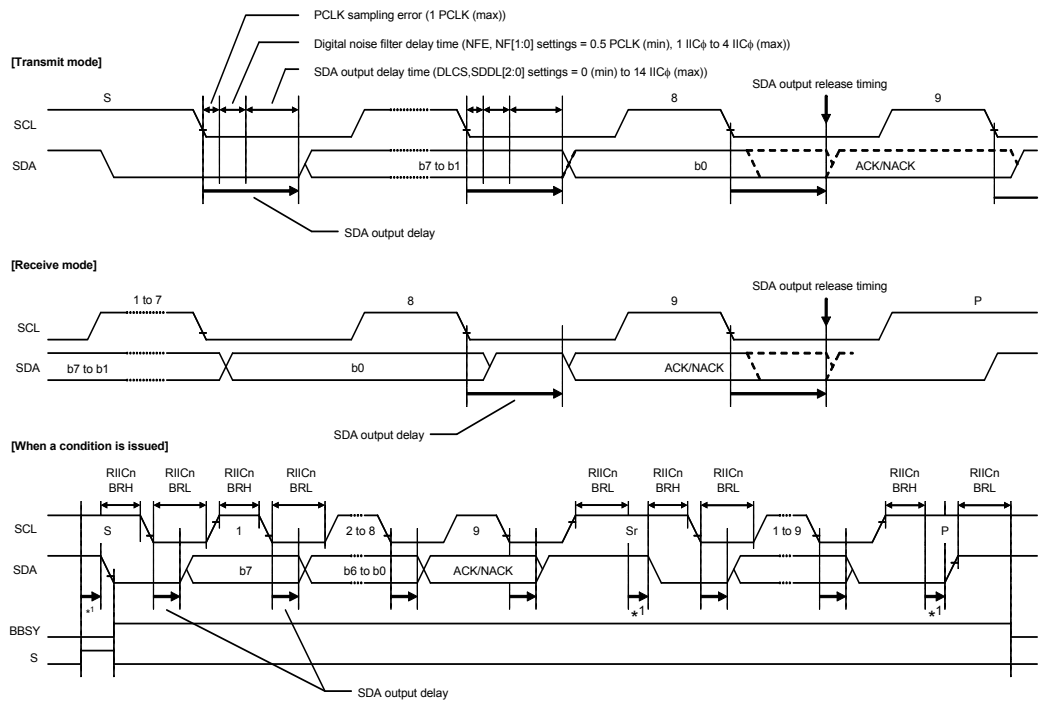
## 20.7 Facility for Delaying SDA Output

The RIIC module incorporates a facility for delaying output on the SDA line. The delay can be applied to all output (issuing of the start, restart, and stop conditions, data, and the ACK and NACK signals) on the SDA line.

With the SDA output delay facility, SDA output is delayed from detection of a falling edge of the SCL signal to ensure that the SDA signal is output within the interval over which the SCL (clock) signal is at the low level. Doing this leads to usage with the aim of preventing erroneous operation of communications devices.

The output delay facility is enabled by setting the RIICnMR2.SDDL[2:0] bits to any value other than $000_B$, and disabled by setting the same bits to $000_B$.

While the SDA output delay facility is enabled (i.e. while the SDDL[2:0] bits are set to any value other than $000_B$), the RIICnMR2.DLCS bit selects the clock source for counting by the SDA output delay counter as the internal base clock (IICϕ) for the RIIC module or as a clock signal derived by dividing the frequency of the internal base clock by two (IICϕ/2). The counter counts the number of cycles set in the SDDL[2:0] bits. After counting of the set number of cycles of delay is completed, the RIIC module places the required output (start, restart, or stop condition, data, or an ACK or NACK signal) on the SDA line.




Note 1. The output delay facility is set by the SDDL[2:0] bits when a start (S), restart (Sr), or stop (P) condition is issued.

**Figure 20.22 SDA Output Delay Facility**

## 20.8 Digital Noise-Filter Circuits

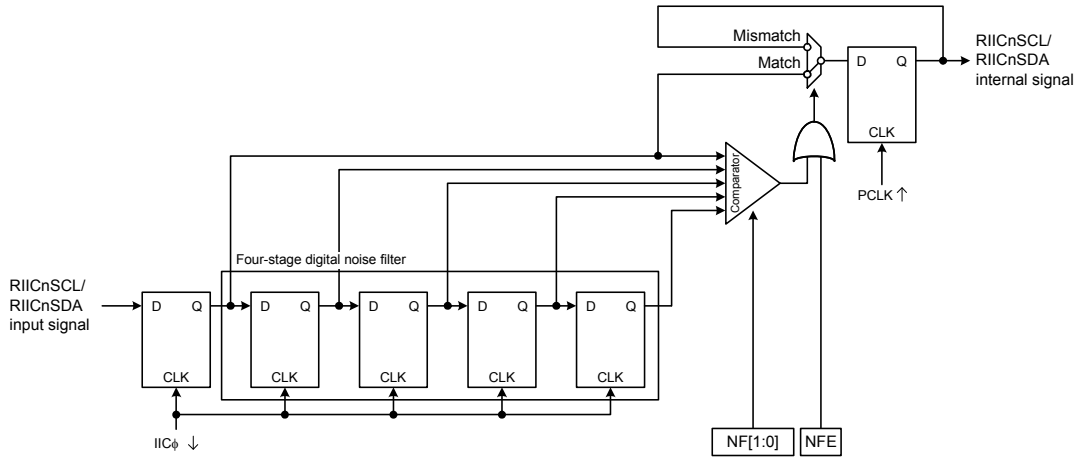
The states of the RIICnSCL and RIICnSDA pins are conveyed to the internal circuitry through the digital noise-filter circuit. **Figure 20.23** is a block diagram of the digital noise-filter circuit.

The on-chip digital noise-filter circuit of the RIIC consists of four flip-flop circuit stages connected in series and a match-detection circuit.

The number of effective stages in the digital noise filter is selected by the RIICnMR3.NF[1:0] bits. The selected number of effective stages determines the noise-filtering capability as a period from one to four IICϕ cycles.

The input signal to the RIICnSCL pin (or RIICnSDA pin) is sampled on falling edges of the IICϕ signal. When the input signal level matches the output level of the number of effective flip-flop circuit stages as selected by the RIICnMR3.NF[1:0] bits, the signal level is conveyed as an internal signal. If the signal levels do not match, the previous value is retained.

If the ratio between the frequency of the internal operating clock (PCLK) and the transfer rate is small, the characteristics of the digital noise filter may lead to the elimination of needed signals as noise. In such cases, it is possible to disable the digital noise-filter circuit (by clearing the RIICnFER.NFE bit to 0).




**Note:** NFE: Digital noise filter circuit enable bit
NF[1:0]: Digital noise filter stage selection bit

**Figure 20.23 Block Diagram of Digital Noise Filter Circuit**

## 20.9 Address Match Detection

The RIIC can set three unique slave addresses in addition to the general call address and device ID address, and also can set 7-bit or 10-bit slave addresses.

### 20.9.1 Slave-Address Match Detection

The RIIC can set three unique slave addresses, and has a slave address detection function for each unique slave address. When the RIICnSER.SARy bit (y = 0 to 2) is set to 1, the slave addresses set in RIICnSARy (y = 0 to 2) can be detected.

When the RIIC detects a match of the set slave address, the corresponding RIICnSR1.AASy flag (y = 0 to 2) is set to 1 at the rising edge of the ninth SCL clock cycle and returns the value set in the RIICnMR3.ACKBT bit to the acknowledge bit on the ninth cycle of SCL clock. The RIICnSR2.RDRF flag or the RIICnSR2.TDRE flag is set to 1 by the following R/W# bit. This causes a receive complete interrupt (INTRIICnRI) or transmit data empty interrupt (INTRIICnTI) to be generated. The AASy flag is used to identify which slave address has been specified.

**Figure 20.24** to **Figure 20.26** show the AASy flag set timing in three cases.

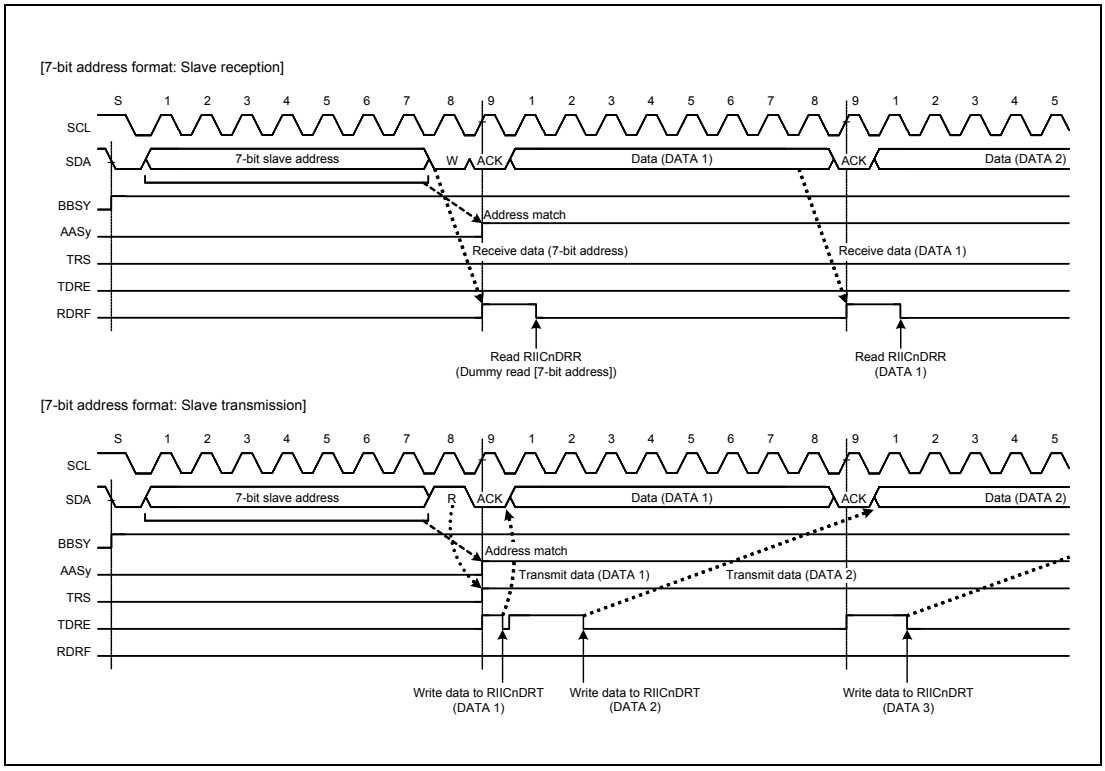



**Figure 20.24 AASy Flag Set Timing with 7-Bit Address Format Selected**

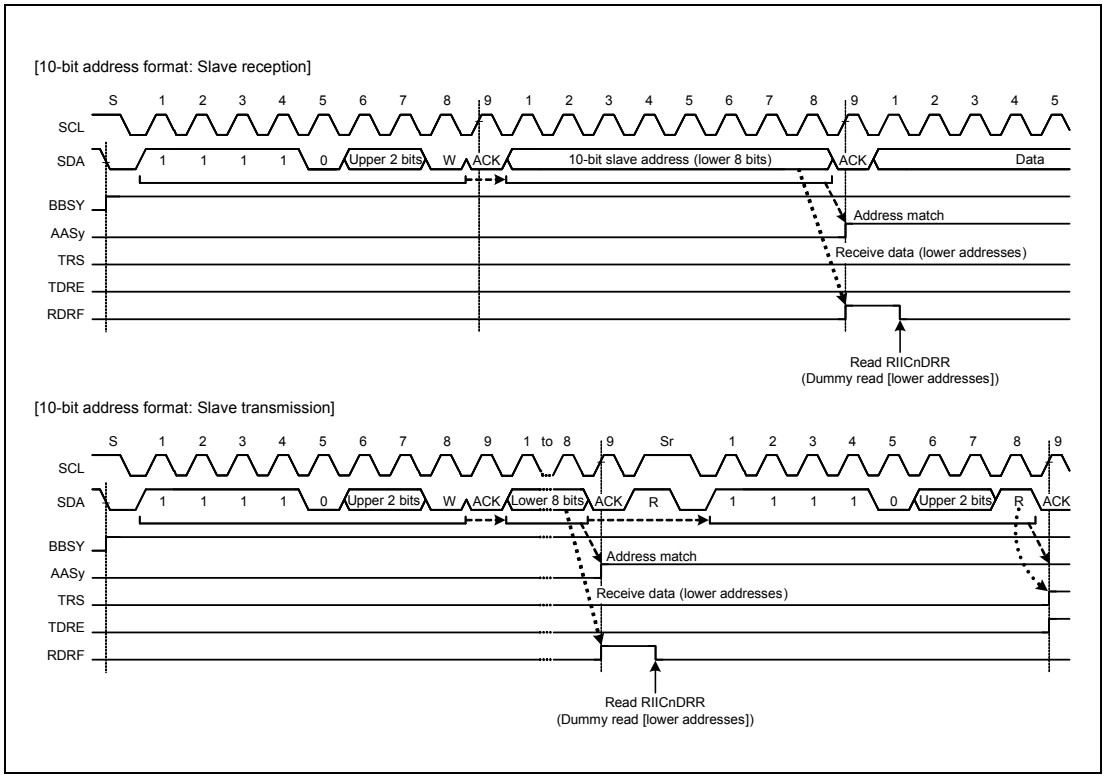



**Figure 20.25 AASy Flag Set Timing with 10-Bit Address Format Selected**

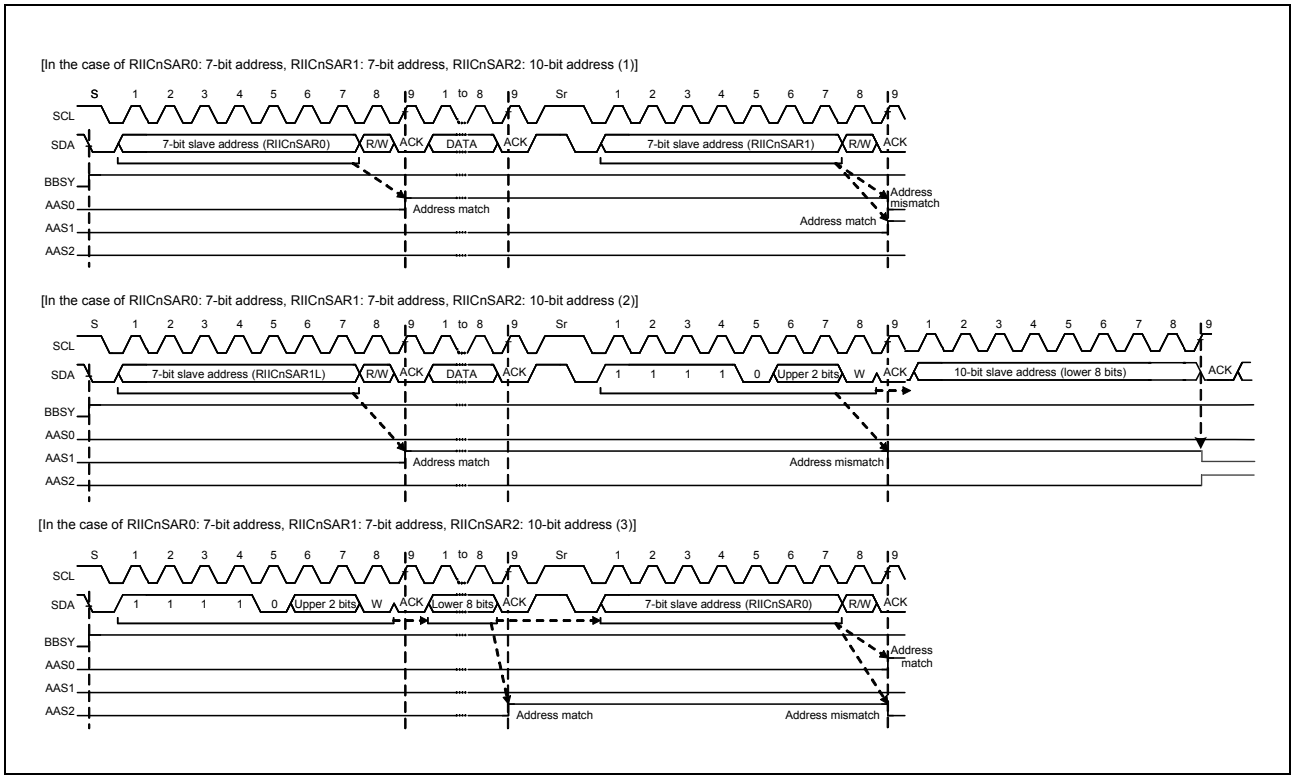



**Figure 20.26 AASy Flag Set/Clear Timing with 7-Bit/10-Bit Address Formats Mixed**

### 20.9.2 Detection of the General Call Address

The RIIC has a facility for detecting the general call address (0000 $000_B$ + 0 [W]). This is enabled by setting the RIICnSER.GCAE bit to 1.

If the address received after a start or restart condition is issued is 0000 $000_B$ + 1[R] (start byte), the RIIC recognizes this as the address of a slave device with an “all-zero” address but not as the general call address.

When the RIIC detects the general call address, both the RIICnSR1.GCA flag and the RIICnSR2.RDRF flag are set to 1 on the rising edge of the ninth cycle of SCL clock. This leads to the generation of a receive complete interrupt (INTRIICnRI). The value of the GCA flag can be confirmed to recognize that the general call address has been transmitted.

Operation after detection of the general call address is the same as normal slave receive operation.

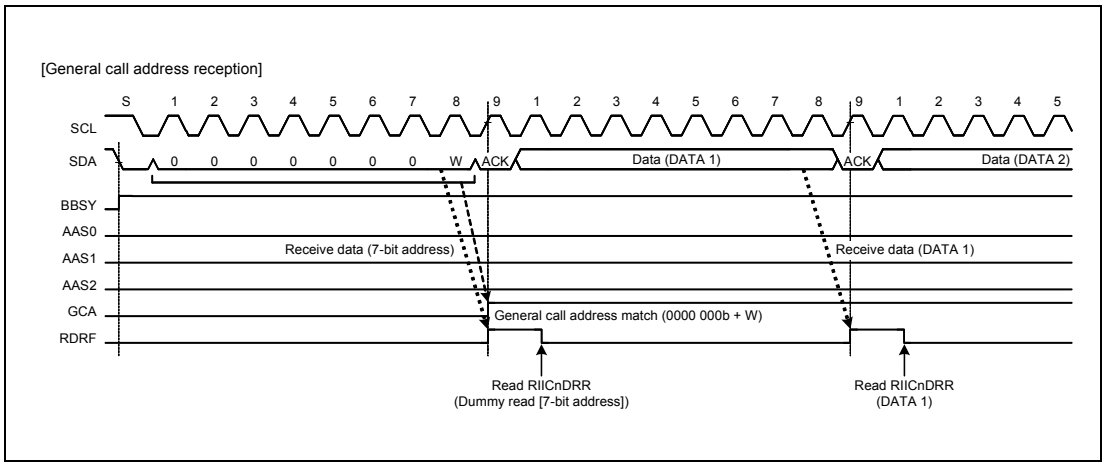



**Figure 20.27 Timing of GCA Flag Setting during Reception of General Call Address**

### 20.9.3 Device-ID Address Detection

The RIIC module has a facility for detecting device-ID addresses conforming with the I2C bus specification (Rev. 03). When the RIIC receives 1111 $100_B$ as the first byte after a start condition or restart condition was issued with the RIICnSER.DIDE bit set to 1, the RIIC recognizes the address as a device ID, sets the RIICnSR1.DID flag to 1 on the rising edge of the ninth SCL clock cycle when the following R/W# bit is 0, and then compares the second and subsequent bytes with its own slave address. If the address matches the value in the slave address register, the RIIC sets the corresponding RIICnSR1.AASy flag (y = 0 to 2) to 1.

After that, when the first byte received after a start or restart condition is issued matches the device ID address (1111 $100_B$) again and the following R/W# bit is 1, the RIIC does not compare the second and subsequent bytes and sets the RIICnSR2.TDRE flag to 1.

In the device-ID address detection function, the RIIC clears the DID flag to 0 if a match with the RIIC's own slave address is not obtained or a match with the device ID address is not obtained after a match with the RIIC's own slave address and the detection of a restart condition. If the first byte after detection of a start or restart condition matches the device ID address (1111 $100_B$) and the R/W# bit is 0, the RIIC sets the DID flag to 1 and compares the second and subsequent bytes with the RIIC's slave address. If the R/W# bit is 1, the DID flag holds the previous value and the RIIC does not compare the second and subsequent bytes. Therefore, the reception of a device-ID address can be checked by reading the DID flag after confirming that TDRE = 1.

Furthermore, prepare the device-ID fields (three bytes: 12 bits indicating the manufacturer + 9 bits identifying the part + 3 bits indicating the revision) that must be sent to the host after reception of a continuous device-ID field as normal data for transmission. For details, see I2C Bus Standard from NXP Semiconductors.

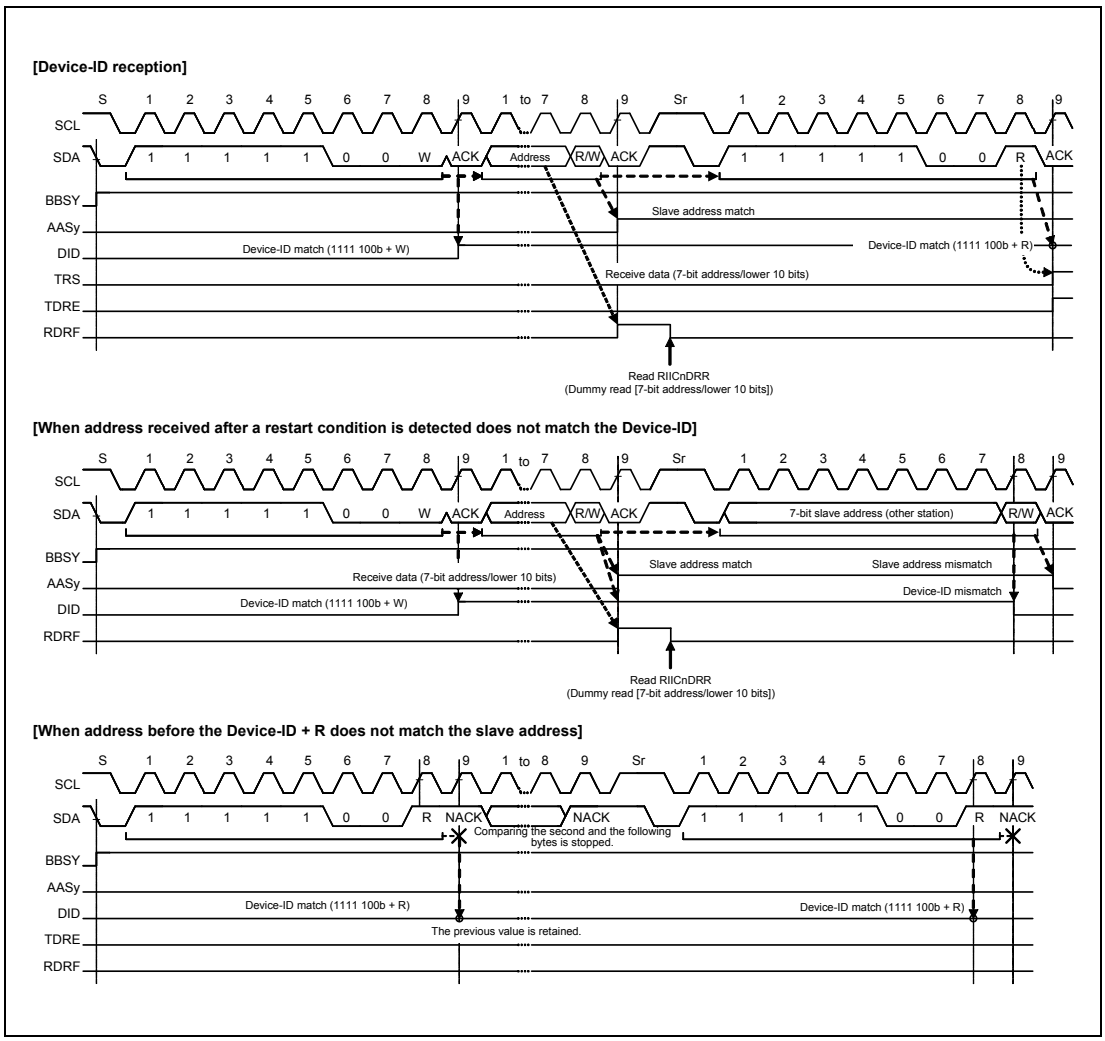



**Figure 20.28 AASy/DID Flag Set/Clear Timing during Reception of Device-ID**

# 20.10 Automatic Low-Hold Function for SCL

## 20.10.1 Function to Prevent Wrong Transmission of Transmit Data

To prevent the unintended transmission of erroneous data, this low-hold period is extended until data for transmission have been written. In addition, the RIIC holds the SCL line low over the period until a stop condition is issued and also over the period until the RIICnDRR register is dummy read.

**<Master transmit mode>**

- Low-level interval after a start condition or restart condition is issued
- Low-level interval between the ninth clock cycle of one transfer and the first clock cycle of the next
- Low-level interval from the ninth clock cycle until a stop condition is issued

**<Slave transmit mode>**

- Low-level interval between the ninth clock cycle of one transfer and the first clock cycle of the next
- Low-level interval from the ninth clock cycle and the RIICnDRR register is dummy read

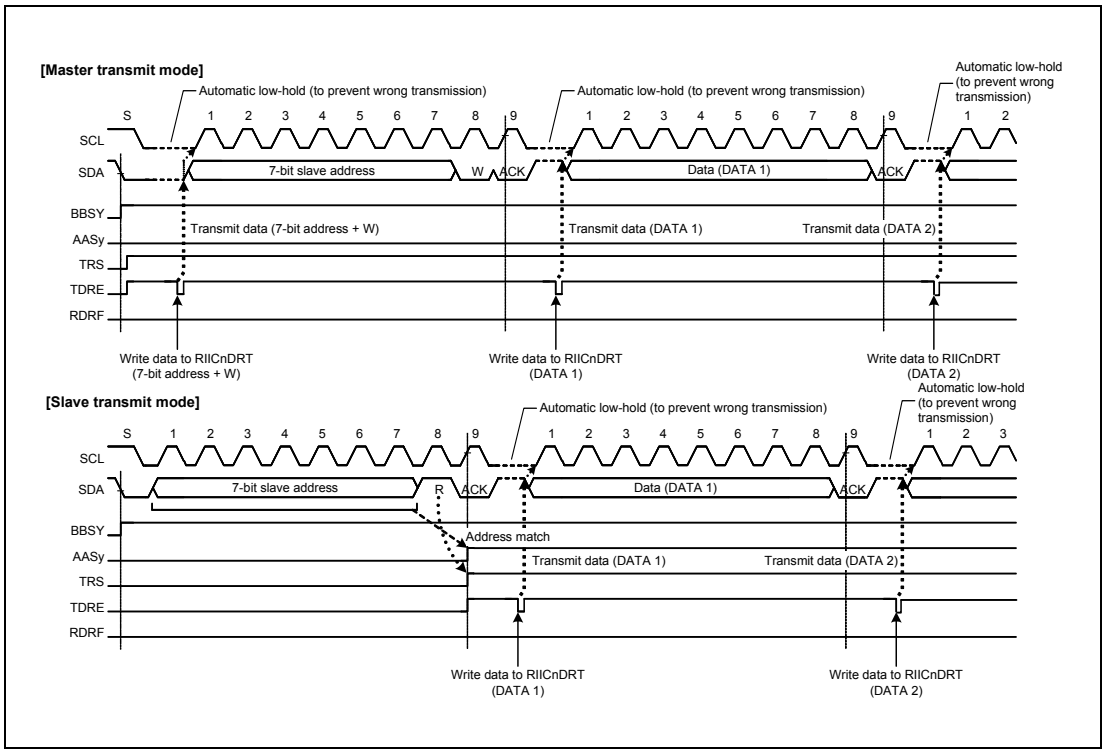



**Figure 20.29 Automatic Low-Hold Operation in Transmit Mode**

## 20.10.2 NACK Reception Transfer Suspension Function

The RIIC has a function to suspend transfer operation when NACK is received in transmit mode (RIICnCR2.TRS bit = 1). This function is enabled when the RIICnFER.NACKE bit is set to 1 (transfer suspension enabled). If the next transmit data has already been written (RIICnSR2.TDRE flag = 0) when NACK is received, next data transmission at the falling edge of the ninth SCL clock cycle is automatically suspended. This prevents the SDA line output level from being held low when the MSB of the next transmit data is 0.

If the transfer operation is suspended by this function (RIICnSR2.NACKF flag = 1), transmit operation and receive operation are discontinued. To restore transmit/receive operation, be sure to clear the NACKF flag to 0. In master transmit mode, clear the NACKF flag to 0, issue a restart or stop condition, and then issue a start condition again.

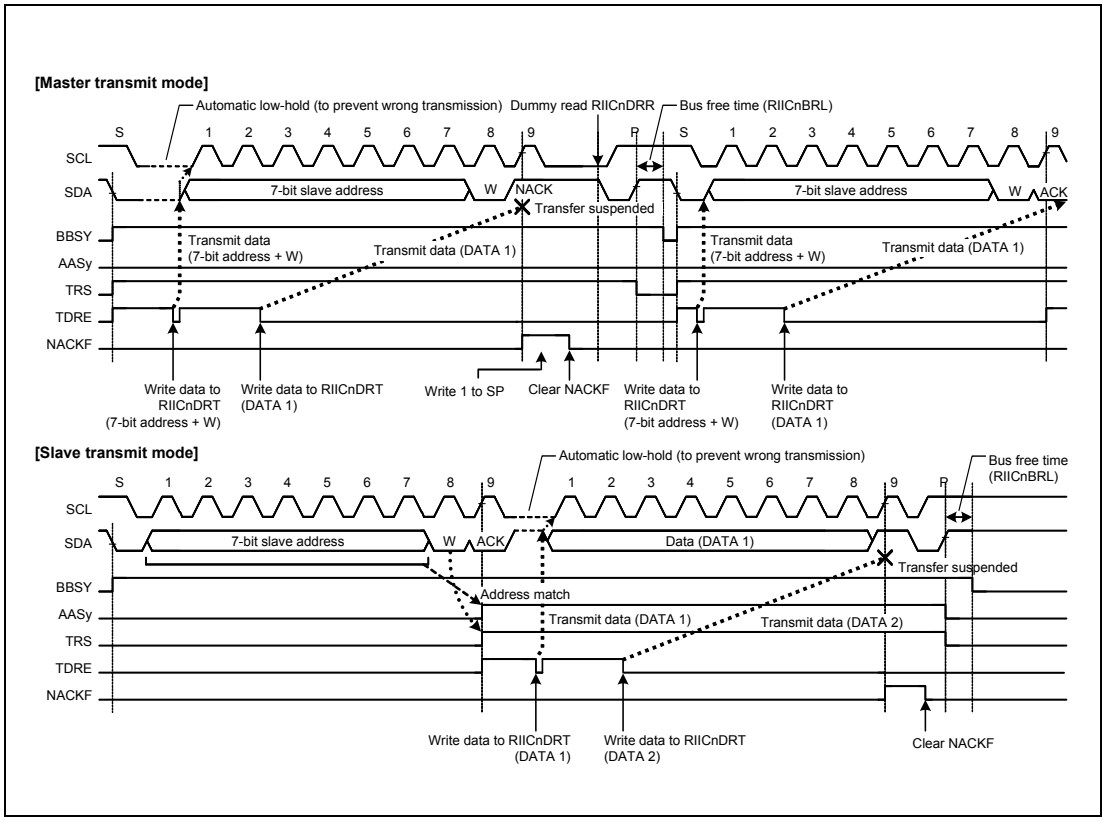



**Figure 20.30 Suspension of Data Transfer when NACK is Received (NACKE = 1)**

### 20.10.3 Function to Prevent Failure to Receive Data

If response processing is delayed when receive data (RIICnDRR) read is delayed for a period of one transfer frame or more with receive complete (RIICnSR2.RDRF flag = 1) in receive mode (RIICnCR2.TRS = 0), the RIIC holds the SCL line low automatically immediately before the next data is received to prevent failure to receive data.

This function to prevent failure to receive data using the automatic low-hold function is also enabled even if the read processing of the final receive data is delayed and, in the meantime, the RIIC's own slave address is designated after a stop condition is issued. This function does not disturb other communication because the RIIC does not hold the SCL line low when a mismatch with its own slave address occurs after a stop condition is issued.

Sections in which the SCL line is held low can be selected with a combination of the RIICnMR3.WAIT and RDRFS bits.

#### (1) One-Byte Receive Operation and Automatic Low-Hold Function Using the WAIT Bit

When the RIICnMR3.WAIT bit is set to 1, the RIIC performs one-byte receive operation using the WAIT bit function.
Furthermore, when the RIICnMR3.RDRFS bit is 0, the RIIC automatically sends the RIICnMR3.ACKBT bit value for the acknowledge bit in the period from the falling edge of the eighth SCL clock cycle to the falling edge of the ninth SCL clock cycle, and automatically holds the SCL line low at the falling edge of the ninth SCL clock cycle using the WAIT bit function. This low-hold is released by reading data from RIICnDRR, which enables bytewise receive operation.

The WAIT bit function is enabled for receive frames after a match with the RIIC's own slave address (including the general call address and device ID address) is obtained in master receive mode or slave receive mode.

#### (2) One-Byte Receive Operation (ACK/NACK Transmission Control) and Automatic Low-Hold Function Using the RDRFS Bit

When the RIICnMR3.RDRFS bit is set to 1, the RIIC performs one-byte receive operation using the RDRFS bit function.
When the RIICnSR2.RDRFS bit is set to 1, the RDRF flag (receive complete) in RIICnSR2 is set to 1 at the rising edge of the eighth SCL clock cycle, and the SCL line is automatically held low at the falling edge of the eighth SCL clock cycle. This lowhold is released by writing a value to the RIICnMR3.ACKBT bit, but cannot be released by reading data from RIICnDRR, which enables receive operation by the ACK/NACK transmission control according to the data received in byte units.

The RDRFS bit function is enabled for receive frames after a match with the RIIC's own slave address (including the general call address and device ID address) is obtained in master receive mode or slave receive mode.

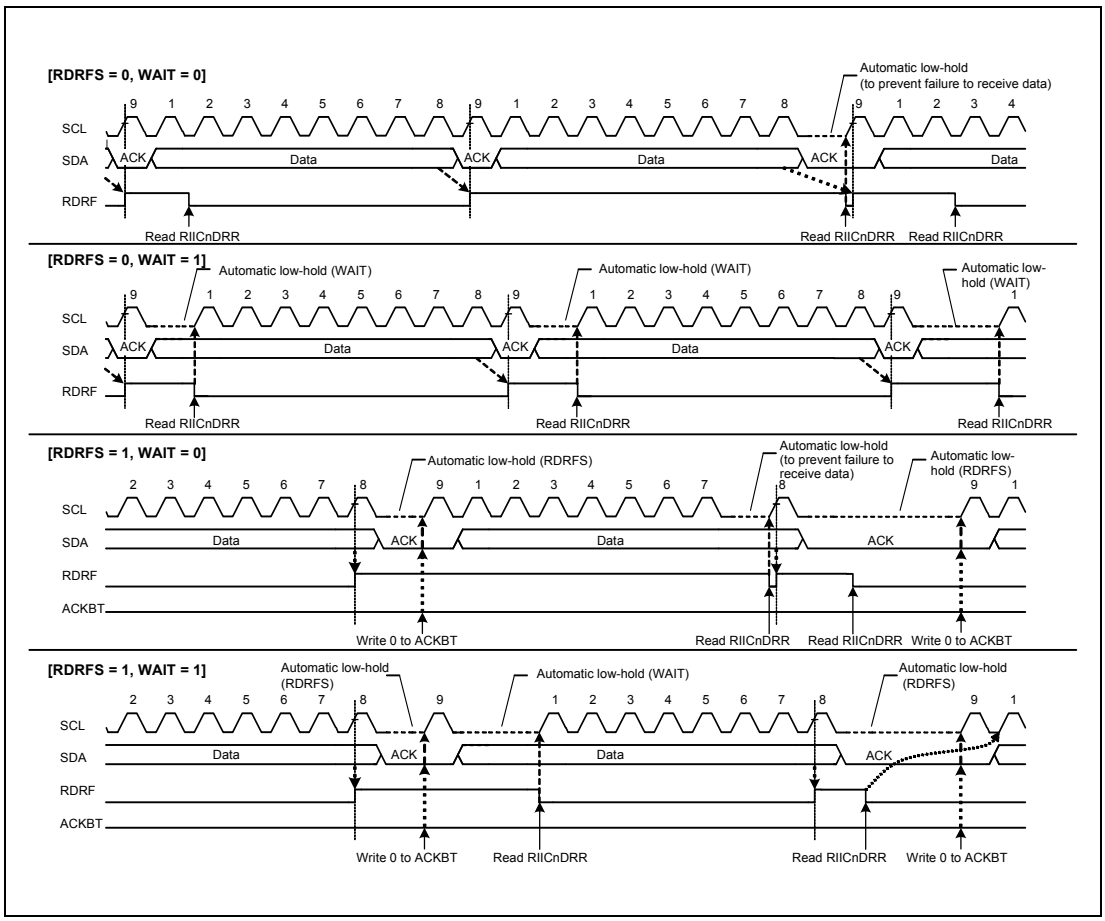



**Figure 20.31 Automatic Low-Hold Operation in Receive Mode (Using RDRFS and WAIT Bits)**

## 20.11 Arbitration-Lost Detection Functions

In addition to the normal arbitration-lost detection function defined by the I2C bus standard, the RIIC has functions to prevent double-issue of a start condition, to detect arbitration-lost during transmission of NACK, and to detect arbitration-lost in slave transmit mode.

### 20.11.1 Master Arbitration-Lost Detection (MALE Bit)

The RIIC drives the SDA line low to issue a start condition. However, if the SDA line has already been driven low by another master device issuing a start condition, the RIIC considers this a loss in arbitration, so priority is given to transfer by the other master device. Similarly, if the RIICnCR2.ST bit is set to 1 while the bus is busy (RIICnCR2.BBSY flag = 1), the RIIC considers itself to have lost in arbitration, so priority is given to transfer by the other master device and no start condition is generated.

When a start condition is issued successfully, if the data for transmission including the address bits (i.e. the internal SDA output level) and the level on the SDA line do not match (the high output as the internal SDA output; i.e. the SDA pin is in the high-impedance state, and the low level is detected on the SDA line), the RIIC loses in arbitration.

After a loss in arbitration of mastership, the RIIC immediately enters slave receive mode. If a slave address (including the general call address) matches its own address at this time, the RIIC continues in slave operation.

A loss in arbitration of mastership is detected when the following conditions are met while the RIICnFER.MALE bit is 1 (master arbitration-lost detection enabled).

[Master arbitration-lost conditions]

- Non-matching of the internal level for output on SDA and the level on the SDA line after a start condition was issued by setting the RIICnCR2.ST bit to 1 while the RIICnCR2.BBSY flag was cleared to 0 (erroneous issuing of a start condition)
- Setting of the RIICnCR2.ST bit to 1 (start condition double-issue error) while the RIICnCR2.BBSY flag is set to 1
- When the transmit data excluding acknowledge (internal SDA output level) does not match the level on the SDA line in master transmit mode (RIICnCR2.MST and TRS bits = $11_B$)

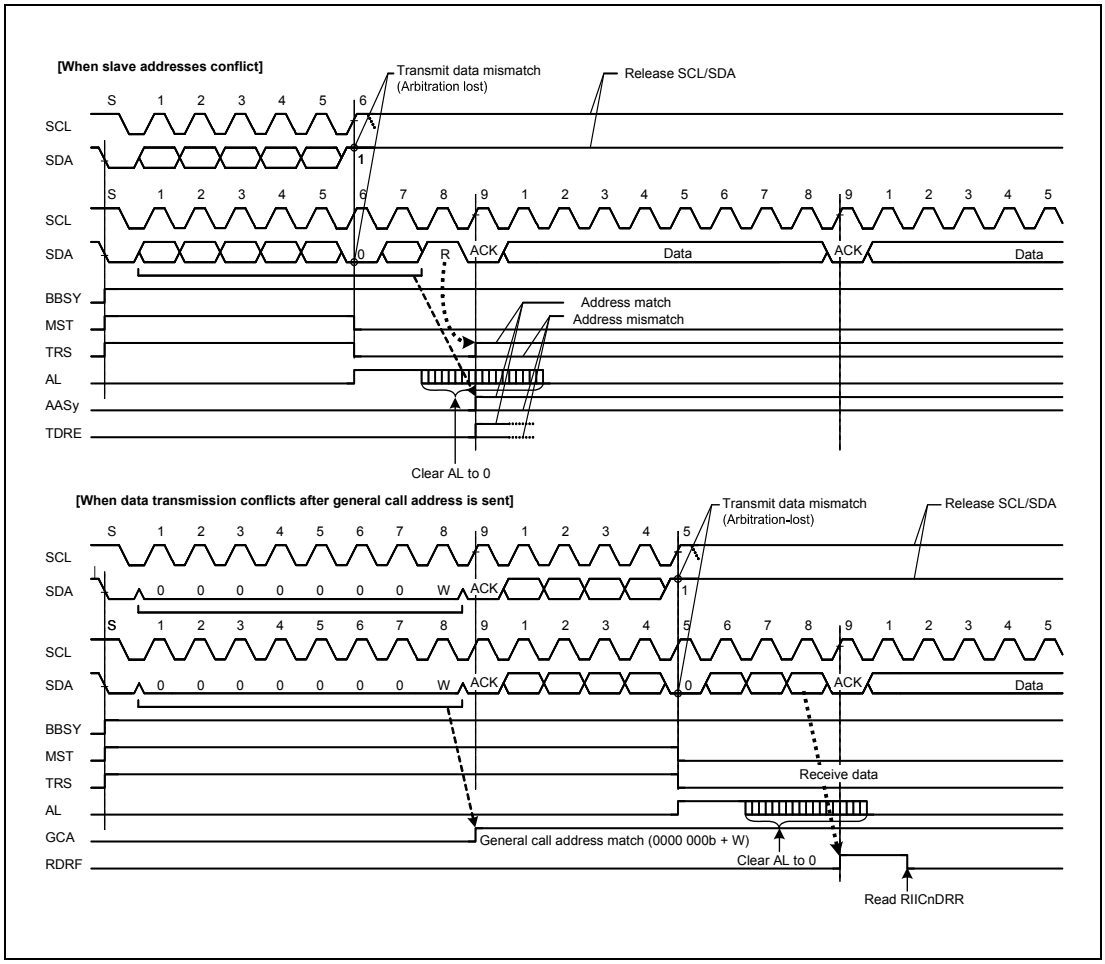



**Figure 20.32 Examples of Master Arbitration-Lost Detection (MALE = 1)**

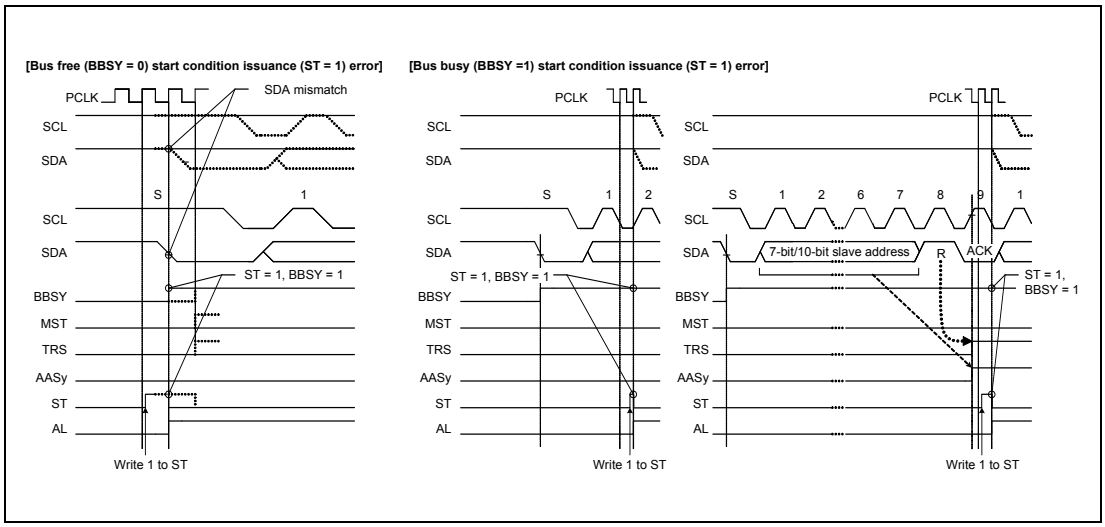



**Figure 20.33 Arbitration-Lost when a Start Condition is Issued (MALE = 1)**

The TRS bit is not cleared if 1 is written to the ST bit while the BBSY flag is 1 in slave transmit mode.

### 20.11.2 Function to Detect Loss of Arbitration during NACK Transmission (NALE Bit)

The RIIC has a function to cause arbitration to be lost if the internal SDA output level does not match the level on the SDA line (the high output as the internal SDA output; i.e. the SDA pin is in the high-impedance state, and the low level is detected on the SDA line) during transmission of NACK in receive mode. Arbitration is lost due to a conflict of NACK transmission and ACK transmission when two or more master devices receive data from the same slave device simultaneously in a multi-master system. Such conflict occurs when multiple master devices send/receive the same information through a single slave device. **Figure 20.34** shows an example of arbitration-lost detection during transmission of NACK.

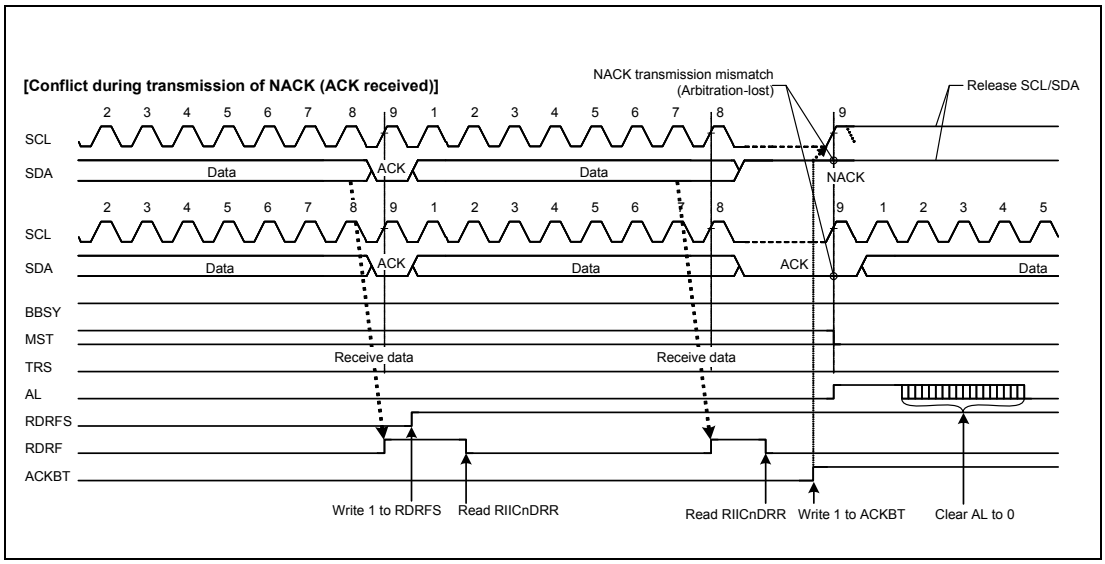



**Figure 20.34 Example of Arbitration-Lost Detection during Transmission of NACK (NALE = 1)**

The following explains arbitration-lost detection using an example where two master devices (master A and master B) and a single slave device are connected through the bus. In this example, master A receives two bytes of data from the slave device, and master B receives four bytes of data from the slave device.

If master A and master B access the slave device simultaneously, because the slave address is identical, arbitration is not lost in both master A and master B during access to the slave device. Therefore, both master A and master B recognize that they have obtained the bus mastership and operate as such. Here, master A sends NACK when it has received two final bytes of data from the slave device. Meanwhile, master B sends ACK because it has not received necessary four bytes of data. At this time, the NACK transmission from master A and the ACK transmission from master B conflict. In general, if a conflict like this occurs, master A cannot detect ACK transmitted by master B and issues a stop condition.

Therefore, the issuance of the stop condition conflicts with the SCL clock output of master B, which disturbs communication.

When the RIIC receives ACK during transmission of NACK, it detects a defeat in conflict with other master devices and causes arbitration to be lost.

If arbitration is lost during transmission of NACK, the RIIC enters slave receive mode. This prevents a stop condition from being issued, preventing a communication failure on the bus.

The RIIC detects arbitration-lost during transmission of NACK when the following condition is met with the RIICnFER.NALE bit set to 1 (arbitration-lost detection during NACK transmission enabled).

[Condition for arbitration-lost during NACK transmission]

When the internal SDA output level does not match the SDA line (ACK is received) during transmission of NACK (RIICnMR3.ACKBT bit = 1)

### 20.11.3 Slave Arbitration-Lost Detection (SALE Bit)

The RIIC has a function to cause arbitration to be lost if the data for transmission (i.e. the internal SDA output level) and the level on the SDA line do not match (the high output as the internal SDA output; i.e. the SDA pin is in the highimpedance state, and the low level is detected on the SDA line) in slave transmit mode.

When it loses slave arbitration, the RIIC enters slave receive mode.

The RIIC detects slave arbitration-lost when the following condition is met with the RIICnFER.SALE bit set to 1 (slave arbitration-lost detection enabled).

[Condition for slave arbitration-lost]

When transmit data excluding acknowledge (internal SDA output level) does not match the SDA line in slave transmit mode (RIICnCR2.MST and TRS bits = $01_B$)

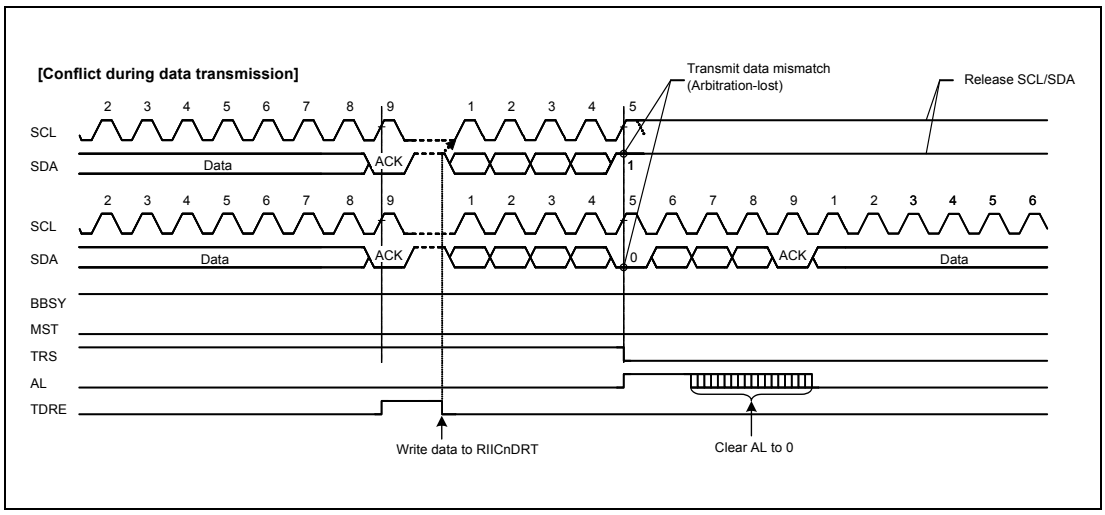



**Figure 20.35 Example of Slave Arbitration-Lost Detection (SALE = 1)**

# 20.12 Start Condition/Restart Condition/Stop Condition Issuing Function

## 20.12.1 Issuing a Start Condition

The RIIC issues a start condition when the RIICnCR2.ST bit is set to 1.

When the ST bit is set to 1, a start condition issuance request is made and the RIIC issues a start condition when the RIICnCR2.BBSY flag is 0 (bus free). When a start condition is issued normally, the RIIC automatically shifts to the master transmit mode.

A start condition is issued in the following sequence.

[Start condition issuance]

- Drive the SDA line low (high level to low level).
- Ensure the time set in RIICnBRH and the start condition hold time.
- Drive the SCL line low (high level to low level).
- Detect low level of the SCL line and ensure the low-level period of SCL line set in RIICnBRL.

## 20.12.2 Issuing a Restart Condition

The RIIC issues a restart condition when the RIICnCR2.RS bit is set to 1.

When the RS bit is set to 1, a restart condition issuance request is made even during communication and the RIIC issues a restart condition when the RIICnCR2.BBSY flag is 1 (bus busy) and the RIICnCR2.MST bit is 1 (master mode).

(To detect the issuance of a restart condition, clear the RIICnSR2.START flag before a restart condition is issued.)

A restart condition is issued in the following sequence.

[Restart condition issuance]

- Release the SDA line.
- Ensure the low-level period of SCL line set in RIICnBRL.
- Release the SCL line (low level to high level).
- Detect a high level of the SCL line and ensure the time set in RIICnBRL and the restart condition setup time.
- Drive the SDA line low (high level to low level).
- Ensure the time set in RIICnBRH and the restart condition hold time.
- Drive the SCL line low (high level to low level).
- Detect a low level of the SCL line and ensure the low-level period of SCL line set in RIICnBRL.

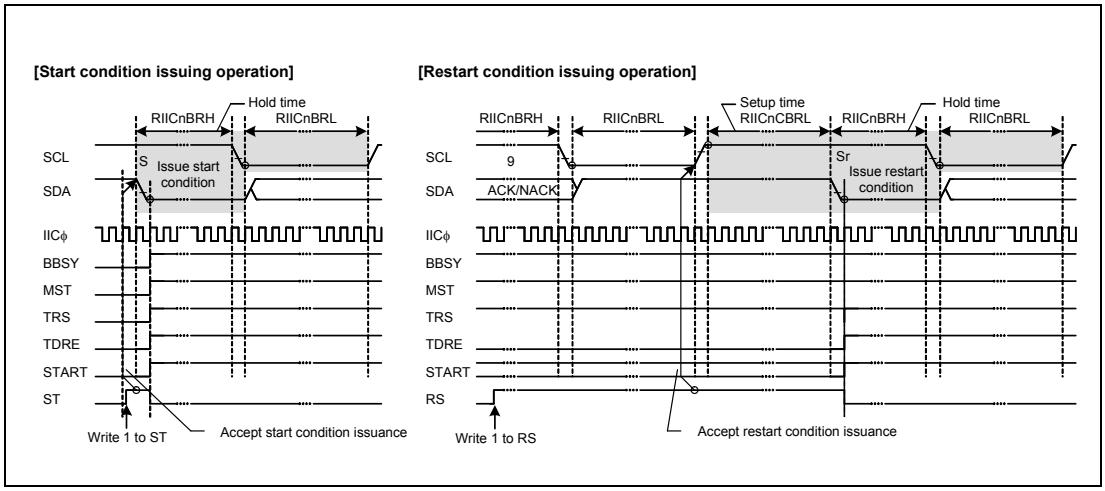



**Figure 20.36 Start Condition/Restart Condition Issue Timing (ST and RS Bits)**

### 20.12.3 Issuing a Stop Condition

The RIIC issues a stop condition when the RIICnCR2.SP bit is set to 1.

When the SP bit is set to 1, a stop condition issuance request is made and the RIIC issues a stop condition when the RIICnCR2.BBSY flag is 1 (bus busy) and the RIICnCR2.MST bit is 1 (master mode).

A stop condition is issued in the following sequence.

[Stop condition issuance]

- Drive the SDA line low (high level to low level).
- Ensure the low-level period of SCL line set in RIICnBRL.
- Release the SCL line (low level to high level).
- Detect a high level of the SCL line and ensure the time set in RIICnBRH and the stop condition setup time.
- Release the SDA line (low level to high level).
- Ensure the time set in RIICnBRL and the bus free time.
- Clear the BBSY flag to 0 (to release the bus mastership).

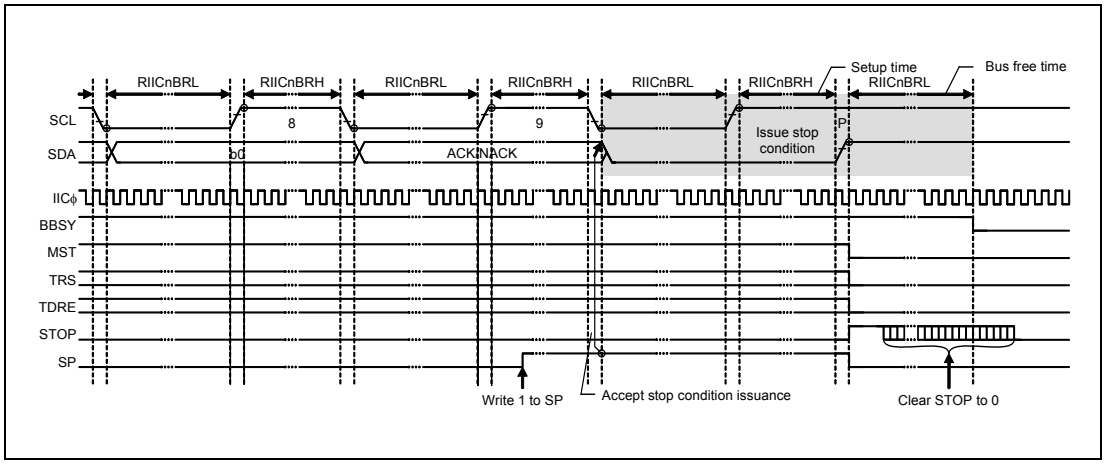



**Figure 20.37 Stop Condition Issue Timing (SP Bit)**

## 20.13 Bus Hanging

If the clock signals from the master and slave devices go out of synchronization due to noise or other factors, the I2C bus might hang with a fixed level on the SCL line and/or SDA line.

As measures against the bus hanging, the RIIC has a timeout function to detect hanging by monitoring the SCL line, a function for the output of an extra SCL clock cycle to release the bus from a hung state due to clock signals being out of synchronization, and the RIIC/internal reset function.

By checking the RIICnCR1.SCLO, SDAO, SCLI, and SDAI bits, it is possible to see whether the RIIC or its partner in communications is placing the low level on the SCL or SDA lines.

### 20.13.1 Timeout Function

The RIIC has the timeout function to detect an abnormality that the SCL line is held for a certain period of time. The RIIC can detect an abnormal bus state by monitoring that the SCL line is held low or high for a predetermined time.

The timeout function monitors the SCL line state and counts the low-level period or high-level period using the internal counter. The timeout function resets the internal counter each time the SCL line changes (rising or falling), but continues to count unless the SCL line changes. If the internal counter overflows due to no SCL line change, the RIIC can detect the timeout and report the bus abnormality.

The internal counter is cleared when one of the conditions is met.

(1) When RIICnMR2.TMOH=0, and RIICnMR2.TMOL=1:
The internal counter is cleared by SCL rising

(2) When RIICnMR2.TMOH=1, and RIICnMR2.TMOL=0:
The internal counter is cleared by SCL falling

(3) When RIICnMR2.TMOH=RIICnMR2.TMOL=1:
The internal counter is cleared by SCL rising or falling

This timeout function is enabled when the RIICnFER.TMOE bit is 1. It detects an abnormal bus state that the SCL line is stuck low or high during the following conditions:

- The bus is busy (RIICnCR2.BBSY flag is 1) in master mode (RIICnCR2.MST bit is 1).
- The RIIC's own slave address matches (RIICnSR1 register is not $00_H$) and the bus is busy (RIICnCR2.BBSY flag is 1) in slave mode (RIICnCR2.MST bit is 0).
- The bus is free (RIICnCR2.BBSY flag is 0) while generation of a START condition is requested (RIICnCR2.ST bit is 1).

The internal counter of the timeout function works using the internal reference clock (IICϕ) set by the RIICnMR1.CKS[2:0] bits as a count source. It functions as a 16-bit counter when long mode is selected (RIICnMR2.TMOS bit = 0) or a 14-bit counter when short mode is selected (TMOS bit = 1).

The SCL line level (low/high or both levels) during which this counter is activated can be selected by the setting of the RIICnMR2.TMOH and TMOL bits. If both TMOL and TMOH bits are cleared to 0, the internal counter does not work.

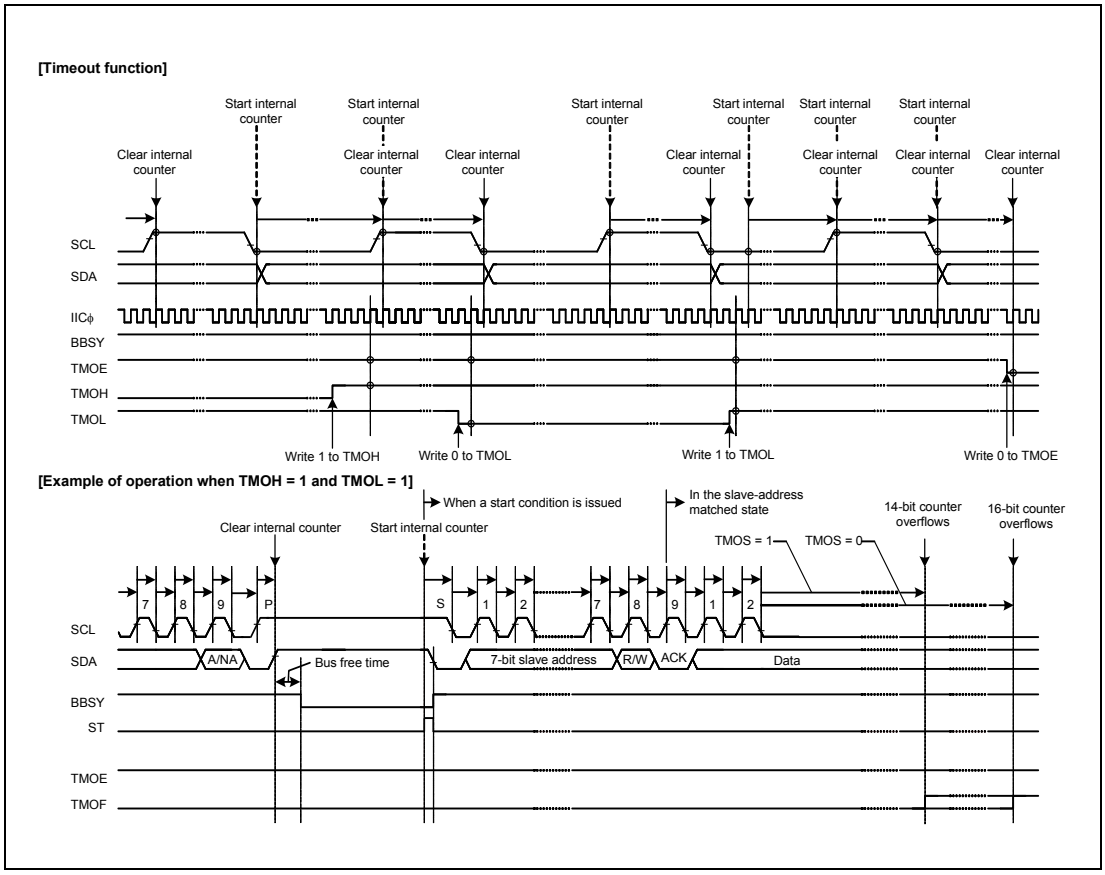



**Figure 20.38 Timeout Function (TMOE, TMOS, TMOH, and TMOL Bits)**

### 20.13.2 Extra SCL Clock Cycle Output Function

In master mode, the RIIC module has a facility for the output of extra SCL (clock) cycles to release the SDA line of the slave device from being held at the low level due to the master being out of synchronization with the slave device.

This function is mainly used in master mode to release the SDA line of the slave device from the state of being fixed to the low level by including extra cycles of SCL output from the RIIC with single cycles of the SCL (clock) signal as the unit in the case of a bus error where the RIIC cannot issue a stop condition because the slave device is holding the SDA line at the low level. Do not use this facility in normal situations. Using it when communications are proceeding correctly will lead to malfunctions.

When the RIICnCR1.CLO bit is set to 1 in master mode, a single cycle of the SCL clock at the frequency corresponding to the transfer rate settings (settings of the RIICnMR1.CKS[2:0] bits, and of the RIICnBRH and RIICnBRL registers) is output as an extra clock cycle. After output of this single cycle of the SCL clock, the CLO bit is automatically cleared to 0. Therefore, further extra clock cycles can be output consecutively by the software program writing 1 to the CLO bit after having read CLO = 0.

When the RIIC module is in master mode and the slave device is holding the SDA line at the low level because synchronization with the slave device has been lost due to the effects of noise, etc., the output of a stop condition is not possible. The facility for output of an extra cycle of the SCL (clock) signal can be used to output extra cycles of SCL one by one to make the slave device release the SDA line from being held at the low level, thus recovering the bus from an unusable state. Release of the SDA line by the slave device can be monitored by reading the RIICnCR1.SDAI bit. After confirming release of the SDA line by the slave device, complete communications by reissuing the stop condition.

Use this facility with the RIICnFER.MALE bit (master arbitration-lost detection disabled) cleared to 0. If the MALE bit is set to 1 (master arbitration-lost detection enabled), arbitration is lost when the value of the RIICnCR1.SDAO bit does not match the state of the SDA line, so take care on this point.

[Additional output conditions for the SCL clock]

- In master mode and when the bus is free
- In master mode and the SCL line is not held low (the bus is busy)

**Figure 20.39** shows the operation timing of the extra SCL clock cycle output function (CLO bit).

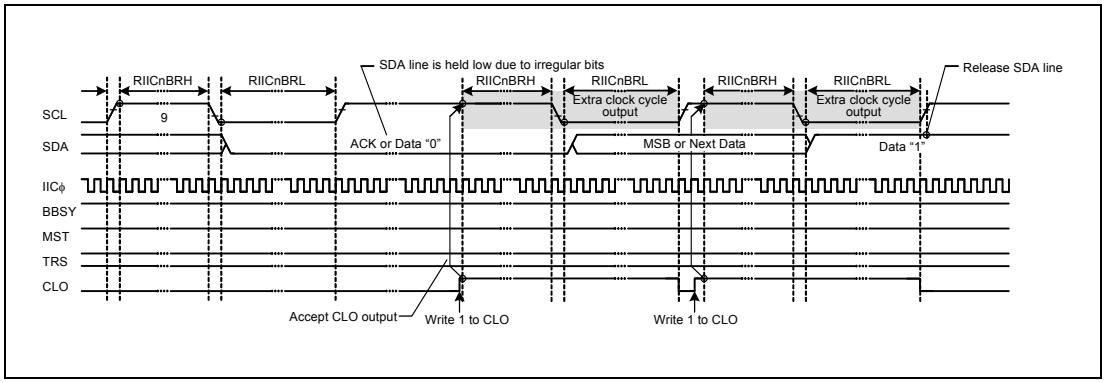



**Figure 20.39 Extra SCL Clock Cycle Output Function (CLO Bit)**

### 20.13.3 RIIC Reset and Internal Reset

The RIIC module incorporates a function for resetting itself. There are two types of reset. One is referred to as an RIIC reset; this initializes all registers including the RIICnCR2.BBSY flag. The other is referred to as an internal reset; this releases the RIIC from the slave-address matched state and initializes the internal counter while retaining other settings.

After issuing a reset, be sure to clear the RIICnCR1.IICRST bit to 0.

Both types of reset are effective for release from bus-hung states since both restore the output state of the SCL and SDA pins to the high impedance state.

Issuing a reset during slave operation may lead to a loss of synchronization between the master device clock and the slave device clock, so avoid this where possible. Note that monitoring of the bus state, such as for the presence of a start condition, is not possible during an RIIC reset (RIICnCR1.ICE and IICRST bits = $01_B$).

For a detailed description of the RIIC and internal resets, see **Section 20.14, Reset Function of RIIC**.

## 20.14 Reset Function of RIIC

The RIIC has RIIC reset, and internal reset functions. In addition RIIC is cleared by ISORES. **Table 20.26** lists the scope of each reset and reset conditions.

**Table 20.26 RIIC Reset Functions (1/2)**

| UM | | ISORES | RIIC Reset (ICE = 0, IICRST = 1) | Internal Reset (ICE = 1, IICRST = 1) | Start/Restart Condition Detection | Stop Condition Detection |
|---|---|---|---|---|---|---|
| RIICnCR1 | ICE | Initialized | 0 | 1 | Retained | Retained |
| | IICRST | Initialized | 1 | 1 | Retained | Retained |
| | CLO | Initialized | Initialized | Retained | Retained | Retained |
| | SOWP | Initialized | Initialized | Retained | Retained | Retained |
| | SCLO | Initialized | Initialized | Initialized | Retained | Retained |
| | SDAO | Initialized | Initialized | Initialized | Retained | Retained |
| | SCLI | Initialized | Initialized | Retained | Retained | Retained |
| | SDAI | Initialized | Initialized | Retained | Retained | Retained |
| RIICnCR2 | BBSY | Initialized | Initialized | Initialized *1 | Operation | Retained |
| | MST | Initialized | Initialized | Initialized | Operation (retained) | Initialized |
| | TRS | Initialized | Initialized | Initialized | Operation (retained) | Initialized |
| | SP | Initialized | Initialized | Initialized | Initialized | Initialized |
| | RS | Initialized | Initialized | Initialized | Initialized | Initialized |
| | ST | Initialized | Initialized | Initialized | Initialized | Retained |
| RIICnMR1 | MTWP | Initialized | Initialized | Retained | Retained | Retained |
| | CKS[2:0] | Initialized | Initialized | Retained | Retained | Retained |
| | BCWP | Initialized | Initialized | Retained | Retained | Retained |
| | BC[2:0] | Initialized | Initialized | Initialized | Initialized | Retained |
| RIICnMR2 | | Initialized | Initialized | Retained | Retained | Retained |
| RIICnMR3 | WAIT | Initialized | Initialized | Retained | Retained | Retained |
| | RDRFS | Initialized | Initialized | Retained | Retained | Retained |
| | ACKWP | Initialized | Initialized | Retained | Retained | Retained |
| | ACKBT | Initialized | Initialized | Retained | Retained | Initialized |
| | ACKBR | Initialized | Initialized | Retained | Retained | Retained |
| | NF[1:0] | Initialized | Initialized | Retained | Retained | Retained |
| RIICnFER | | Initialized | Initialized | Retained | Retained | Retained |
| RIICnSER | | Initialized | Initialized | Retained | Retained | Retained |
| RIICnIER | | Initialized | Initialized | Retained | Retained | Retained |
| RIICnSR1 | DID | Initialized | Initialized | Initialized | Retained | Initialized |
| | GCA | Initialized | Initialized | Initialized | Retained | Initialized |
| | AAS2 | Initialized | Initialized | Initialized | Retained | Initialized |
| | AAS1 | Initialized | Initialized | Initialized | Retained | Initialized |
| | AAS0 | Initialized | Initialized | Initialized | Retained | Initialized |
| RIICnSR2 | TDRE | Initialized | Initialized | Initialized | Retained | Initialized |
| | TEND | Initialized | Initialized | Initialized | Retained | Initialized |
| | RDRF | Initialized | Initialized | Initialized | Retained | Retained |
| | NACKF | Initialized | Initialized | Initialized | Retained | Retained |
| | STOP | Initialized | Initialized | Initialized | Retained | Operation |
| | START | Initialized | Initialized | Initialized | Operation | Initialized |
| | AL | Initialized | Initialized | Initialized | Retained | Retained |
| | TMOF | Initialized | Initialized | Initialized | Retained | Retained |
| RIICnSAR0, 1, 2 | | Initialized | Initialized | Retained | Retained | Retained |

Table 20.26 RIIC Reset Functions (2/2)

| UM | ISORES | RIIC Reset (ICE = 0, IICRST = 1) | Internal Reset (ICE = 1, IICRST = 1) | Start/Restart Condition Detection | Stop Condition Detection |
|---|---|---|---|---|---|
| RIICnBRH, RIICnBRL | Initialized | Initialized | Retained | Retained | Retained |
| RIICnDRT | Initialized | Initialized | Retained | Retained | Retained |
| RIICnDRR | Initialized | Initialized | Retained | Retained | Retained |
| RIICnDRS | Initialized | Initialized | Initialized | Retained | Retained |

Note 1. When an internal reset is applied while the bus is free after detection of a stop condition, the setting of the BBSY flag is 0 while the bus is free following de-assertion of the internal reset signal.
When an internal reset is applied while the bus is not free, the BBSY flag is not cleared to 0.

# Section 21 CAN Interface (RS-CAN)

This section contains a generic description of the CAN Interface (RS-CAN).

A CAN FD interface is also available, but only in PREMIUM models. In ECO and ADVANCED models, the CAN FD channels can be used as classic (non-FD) CAN channels.

The first part of this section describes all RH850/F1K specific properties, such as the number of units, register base addresses, etc. The remainder of the section describes the functions and registers of RS-CANFD and RS-CAN.

## 21.1 Features of RH850/F1K RS-CANFD

### 21.1.1 Number of Units and Channels

This product supports a CAN FD interface that complies with the new ISO/CD 11898-1 standard that revises the CRC field definition to include the stuff bit count. This product can also support the CRC field definition that complies with the ISO/CD 11898-1 standard issued on August 12, 2014 by setting OPBT0.CANFDCRC = 0. For details about OPBT0, see **Section 37, Flash Memory**.

This microcontroller has the following number of RS-CANFD and RS-CAN units.

**Table 21.1 Number of RS-CANFD Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of Units | 1 | | |
| Name | RSCAN0 | | |

**Table 21.2 Number of RS-CAN Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of Units | — | — | 1 |
| Name | — | — | RSCAN1 |

Each unit has the following number of CAN FD interface channels.

**Table 21.3 Unit Configurations and Channels**

| Unit Name | Channel Name | CAN FD Support*1 | RH850/F1K 100 pins (6 ch) | RH850/F1K 144 pins (6 ch) | RH850/F1K 176 pins (7 ch) |
|---|---|---|---|---|---|
| RSCAN0 | CAN0 | Yes | √ | √ | √ |
| | CAN1 | Yes | √ | √ | √ |
| | CAN2 | Yes | √ | √ | √ |
| | CAN3 | Yes | √ | √ | √ |
| | CAN4 | Yes | √ | √ | √ |
| | CAN5 | Yes | √ | √ | √ |
| RSCAN1 | CAN6 | No | — | — | √ |

Note 1. The CAN FD interface is only available in PREMIUM models. In ECO and ADVANCED models, the CAN FD channels can be used as classic (non-FD) CAN channels.

For details about the functions and registers of CAN0 to CAN5 (RS-CANFD), see **Section 21.2, RS-CANFD**.

For details about the functions and registers of CAN6 (RS-CAN), see **Section 21.14, RS-CAN**.

## 21.2 RS-CANFD

This section describes the functions and registers of RS-CANFD.

The RS-CANFD has two interface modes (classical CAN mode and CAN FD mode) and uses different registers for each mode. There are two types of register names RSCANnXXX and RSCFDnCFDXXX (XXX: arbitrary) depending on the interface mode. When explaining specifications common to two registers, register names are described as RSCANn(CFD)XXX.

**Table 21.4 Indices**

| Index | Description |
|---|---|
| n | Throughout this section, the individual RS-CANFD units are generically indicated by the index “n” (n = 0); for example, RSCANn(CFD)GCTR is the global control register of the RSCANn unit. |
| m | Throughout this section, the individual channels of RS-CANFD units are generically indicated by the index “m” (m = 0 to 5); for example, RSCANn(CFD)CmSTS is the channel m status register. |
| j | The individual registers associated with receive rule table are generically indicated by the index “j” (j = 0 to 15); for example, RSCANn(CFD)GAFLIDj is the receive rule ID register. |
| k | The individual transmit/receive FIFO buffers are generically indicated by the index “k” (k = 0 to [channel m × 3 + 2]); for example, RSCANn(CFD)CFCCk is the transmit/receive FIFO buffer configuration/control register. |
| x | The individual receive FIFO buffers are generically identified by the index “x” (x = 0 to 7); for example, RSCANn(CFD)RFSTSx is the receive FIFO buffer status register. |
| d | Data field registers of transmit/receive FIFO buffers and receive FIFO buffers are identified by “d” (classical CAN mode: d = 0 to 1, CAN FD mode: d = 0 to 15). For example, the transmit/receive FIFO buffer data field register is described as RSCANn(CFD)CFDFdk. |
| q | The individual receive buffers are generically indicated by the index “q” (q = 0 to [channel m × 16 + 15]); for example, RSCANn(CFD)RMIDq is the receive buffer ID register. |
| p | The individual transmit buffers are generically indicated by the index “p” (p = 0 to [channel m × 16 + 15]; for example, RSCANn(CFD)TMCp is the transmit buffer control register. |
| b | Data field registers of receive buffers and transmit buffers are identified by “b” (classical CAN mode: b = 0 to 1, CAN FD mode: b = 0 to 4). For example, the receive buffer data field register is described as RSCANn(CFD)RMDFbq. |
| r | The individual RAM tests for CAN are generically indicated by the index “r” (r = 0 to 63); for example, RSCANn(CFD)RPGACCr is the RAM test page access register. |
| y | The registers not covered above are indicated by the index “y” (y = 0 to 2); for example, RSCANn(CFD)RMNDy is a receive buffer new data register. |

**Note:** The functions and descriptions of registers in this section are for a 6-channel RS-CANFD (m = 0 to 5). When referring to information with indices, regard the index values as the ones corresponding to your product. Also, note that, if the value of an index exceeds the range described in this section due to your product, write the value after reset when writing to bits outside the index range.

The following table lists the values of indices for individual products.

**Table 21.5 Indices for Individual Products**

| Indices for Individual Products |
|---|
| **100, 144, 176 pins (6 ch)** |
| j = 0 to 15 |
| k = 0 to 17 |
| x = 0 to 7 |
| q = 0 to 95 |
| p = 0 to 95 |
| r = 0 to 63 |
| y = 0 to 2 |

### 21.2.1 Register Base Address

RSCAN0 base addresses are listed in the following table.

RSCAN0 register addresses are given as offsets from the base addresses.

**Table 21.6 Register Base Addresses**

| Base Address Name | Base Address |
|---|---|
| <RSCAN0_base> | FFD0 $0000_H$ |
| <ECCCAN0_base> for MB RAM | FFC7 $1300_H$ |
| <ECCCANFD0_base> for AFL RAM | FFC7 $1400_H$ |

### 21.2.2 Clock Supply

The RSCANn clock supply is shown in the following table.

**Table 21.7 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| RSCANn (n =0) | clk_xincan | CKSCLK_ICANOSC | Communicatiaon clock from OSC clock |
| | clkc | CKSCLK_IPERI2 | Communication clock |
| | pclk | CKSCLK_ICAN | Module clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CKSCLK_ICAN | |

The operating frequency of the RSCANn depends on the transfer rate and the number of channels in use. Use the RSCANn within the frequency range shown in **Table 21.8**.

**Table 21.8 Range of Operating Frequency Depending on the Transfer Rate and the Number of Channels in Use in the RH850/F1K (1/2)**

| Condition | | Range of Operating Frequency | | |
|---|---|---|---|---|
| Transfer Rate | No. of Channels in Use | pclk | clk_xincan*1 | clkc*1, *2 |
| 1 Mbps | 6ch | pclk ≥ 53MHz | 8 MHz ≤ clk_xincan ≤ pclk/2 | 12.5 MHz ≤ clkc ≤ pclk/2 |
| | 5ch | pclk ≥ 46MHz | | |
| | 4ch | pclk ≥ 40MHz | | |
| | 3ch | pclk ≥ 32MHz | | |
| | 2ch | pclk ≥ 26MHz | | |
| | 1ch | pclk ≥ 18MHz | | |
| 500 kbps | 6ch | pclk ≥ 27MHz | 4 MHz ≤ clk_xincan ≤ pclk/2 | 12.5 MHz ≤ clkc ≤ pclk/2 |
| | 5ch | pclk ≥ 23MHz | | |
| | 4ch | pclk ≥ 20MHz | | |
| | 3ch | pclk ≥ 16MHz | | |
| | 2ch | pclk ≥ 13MHz | | |
| | 1ch | pclk ≥ 8MHz | | |

Table 21.8 Range of Operating Frequency Depending on the Transfer Rate and the Number of Channels in Use in the RH850/F1K (2/2)

| Condition | | Range of Operating Frequency | | |
|---|---|---|---|---|
| Transfer Rate | No. of Channels in Use | pclk | clk_xincan*1 | clkc*1, *2 |
| 125 kbps | 6ch | pclk ≥ 8MHz | 4 MHz ≤ clk_xincan ≤ pclk/2 | 12.5 MHz ≤ clkc ≤ pclk/2 |
| | 5ch | | | |
| | 4ch | | | |
| | 3ch | | | |
| | 2ch | | | |
| | 1ch | | | |

Note 1. Setting the DCS bit in the RSCANn(CFD)GCFG register enables to select either clk_xincan or clkc. Set clocks less than or equal to pclk/2 (up to 40 MHz).

Note 2. When pclk ≤ 25 MHz, select clk_xincan.

**CAUTION**

**When the RS-CANFD module is used in STOP mode, set the MainOSC as the clock source of the RS-CANFD module. For details about how to set the clock source, see Section 12.4.3.10, RS-CAN Clock Domains C_ISO_CAN and C_ISO_CANOSC.**

## 21.2.3 Interrupt Requests

RSCANn interrupt requests are listed in the following table.

Table 21.9 Interrupt Requests (1/2)

| Unit Interrupt Signal | | Description | Interrupt Number | DMA Trigger Number (Only in CAN FD Mode) |
|---|---|---|---|---|
| **RSCANn (n = 0)** | | | | |
| Global | INTRCANGERR0 | CAN global error interrupt | 22 | — |
| | INTRCANGRECC0 | CAN receive FIFO interrupt | 23 | — |
| | RSCANFDRF0 | Reception FIFO Access Message Buffers0 | — | 60 |
| | RSCANFDRF1 | Reception FIFO Access Message Buffers1 | — | 61 |
| | RSCANFDRF2 | Reception FIFO Access Message Buffers2 | — | 62 |
| | RSCANFDRF3 | Reception FIFO Access Message Buffers3 | — | 63 |
| | RSCANFDRF4 | Reception FIFO Access Message Buffers4 | — | 68 |
| | RSCANFDRF5 | Reception FIFO Access Message Buffers5 | — | 69 |
| | RSCANFDRF6 | Reception FIFO Access Message Buffers6 | — | 82 |
| | RSCANFDRF7 | Reception FIFO Access Message Buffers7 | — | 83 |
| CAN0 | INTRCAN0ERR | CAN0 error interrupt | 24 | — |
| | INTRCAN0REC | CAN0 transmit/receive FIFO receive complete interrupt | 25 | — |
| | INTRCAN0TRX | CAN0 transmit interrupt | 26 | — |
| | RSCANFDCF0 | CAN0 Common FIFO Access Message Buffers | — | 23 |
| CAN1 | INTRCAN1ERR | CAN1 error interrupt | 113 | — |
| | INTRCAN1REC | CAN1 transmit/receive FIFO receive complete interrupt | 114 | — |
| | INTRCAN1TRX | CAN1 transmit interrupt | 115 | — |

Table 21.9 Interrupt Requests (2/2)

| Unit Interrupt Signal | | Description | Interrupt Number | DMA Trigger Number (Only in CAN FD Mode) |
|---|---|---|---|---|
| CAN1 | RSCANFDCF1 | CAN1 Common FIFO Access Message Buffers | — | 24 |
| CAN2 | INTRCAN2ERR | CAN2 error interrupt | 217 | — |
| | INTRCAN2REC | CAN2 transmit/receive FIFO receive complete interrupt | 218 | — |
| | INTRCAN2TRX | CAN2 transmit interrupt | 219 | — |
| | RSCANFDCF2 | CAN2 Common FIFO Access Message Buffers | — | 26 |
| CAN3 | INTRCAN3ERR | CAN3 error interrupt | 220 | — |
| | INTRCAN3REC | CAN3 transmit/receive FIFO receive complete interrupt | 221 | — |
| | INTRCAN3TRX | CAN3 transmit interrupt | 222 | — |
| | RSCANFDCF3 | CAN3 Common FIFO Access Message Buffers | — | 27 |
| CAN4 | INTRCAN4ERR | CAN4 error interrupt | 272 | — |
| | INTRCAN4REC | CAN4 transmit/receive FIFO receive complete interrupt | 273 | — |
| | INTRCAN4TRX | CAN4 transmit interrupt | 274 | — |
| | RSCANFDCF4 | CAN4 Common FIFO Access Message Buffers | — | 48 |
| CAN5 | INTRCAN5ERR | CAN5 error interrupt | 287 | — |
| | INTRCAN5REC | CAN5 transmit/receive FIFO receive complete interrupt | 288 | — |
| | INTRCAN5TRX | CAN5 transmit interrupt | 289 | — |
| | RSCANFDCF5 | CAN5 Common FIFO Access Message Buffers | | 49 |

**NOTE**

For the wake-up factors from standby mode, see **Section 14.1.2.1, Wake-Up Factors for Stand-By Modes**.

### 21.2.4 Reset Factors

RSCANn reset factors are listed in the following table. RSCANn is initialized by these reset factors.

**Table 21.10 Reset Factors**

| Unit Name | Reset Factor |
|---|---|
| RSCANn (n = 0) | All reset factors (ISORES) |

### 21.2.5 External Input/Output Signals

External input/output signals of RSCANn are listed below.

**Table 21.11 External Input/Output Signals**

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **RSCANn (n = 0)** | | |
| CANmRX (m = 0 to 5) | CANm receive data input | CANmRX (m = 0 to 5) |
| CANmTX (m = 0 to 5) | CANm transmit data output | CANmTX (m = 0 to 5) |

# 21.3 Overview

## 21.3.1 Functional Overview

The RH850/F1K incorporates one CAN interface unit (RS-CAN) which consists of six channels (CAN0 to CAN 5) of CAN controllers conforming to the ISO11898-1 specification.

**Table 21.12** shows the RS-CANFD module specifications. **Figure 21.1** shows the RS-CANFD module block diagram.

**Table 21.12 RS-CANFD Module Specifications (1/3)**

| Item | Specification |
|---|---|
| Number of channels | 6 |
| Protocol | ISO11898-1 compliant<br>Using CAN FD frames is selectable by switching interface modes. |
| Communication speed | Classical CAN mode:<br>• Maximum 1 Mbps<br>$\text{Communication speed (CANm bit time clock)} = \frac{1}{\text{CANm bit time}}$<br>CANm bit time = CANmTq × Tq count per bit<br>$\text{CANmTq} = \frac{\text{(BRP[9:0] bits in the RSCANnCmCFG register + 1)}}{\text{fCAN}}$<br>fCAN: Frequency of CAN clock (selected by the DCS bit in the RSCANnGCFG register)<br>CAN FD mode:<br>• Nominal bit rate: max.1 Mbps, data bit rate: max. 5 Mbps<br>$\text{Transmission rate (CANm nominal bit time clock)} = \frac{1}{\text{CANm nominal bit time}}$<br>$\text{Transmission rate (CANm data bit time clock)} = \frac{1}{\text{CANm data bit time}}$<br>CANm nominal bit time = CANmTq(N) × Tq count per nominal bit<br>CANm data bit time = CANmTq(D) × Tq count per data bit<br>$\text{CANmTq(N)} = \frac{\text{(NBRP[9:0] bits in the RSCFDnCFDCmNCFG register + 1)}}{\text{fCAN}}$<br>$\text{CANmTq(D)} = \frac{\text{(DBRP[7:0] bits in the RSCFDnCFDCmDCFG register + 1)}}{\text{fCAN}}$<br>fCAN: Frequency of CAN clock (selected by the DCS bit in the RSCFDnCFDGCFG register)<br>m = 0 to 5<br>Tq: Time quantum |
| Buffer | 480 buffers in total<br>• Channel-dedicated: 96 buffers (16 buffers × 6 channels)<br>Transmit buffer: 16 buffers per channel<br>Transmit queue: Single queue per channel (shared with the transmit buffer; up to 16 buffers allocatable)<br>• Shared between channels: 384 buffers for all channels<br>Receive buffer: 0 to 96 buffers<br>Receive FIFO buffer: 8 FIFO buffers (up to 128 buffers allocatable to each)<br>Transmit/receive FIFO buffer: 3 FIFO buffers per channel (up to 128 buffers allocatable to each)<br>• ECC included |

Table 21.12 RS-CANFD Module Specifications (2/3)

| Item | Specification |
|---|---|
| Reception function | • Receives data frames and remote frames.<br>• Selects ID format (standard ID, extended ID, or both IDs) to be received.<br>• Sets interrupt enable/disable for each FIFO.<br>• Mirror function (reception of messages transmitted from the own CAN node)<br>• Timestamp function (to record message reception time as a 16-bit timer value) |
| Reception filter function | • Selects receive messages according to 384 receive rules.<br>• Sets the number of receive rules (0 to 128) for each channel.<br>• Acceptance filter processing: Sets ID and mask for each receive rule.<br>• DLC filter processing: Enables DLC filter check for each acceptance rule. |
| Receive message transfer function | • Routing function<br>Transfers receive messages to arbitrary destinations (can be transferred to up to 8 buffers)<br>Transfer destination: Receive buffer, receive FIFO buffer, and/or transmit/receive FIFO buffer<br>• Labeling function<br>Stores label information together with a message in a receive buffer and FIFO buffer. |
| Transmission function | • Transmits data frames and remote frames.<br>• Selects ID format (standard ID, extended ID, or both IDs) to be transmitted.<br>• Sets interrupt enable/disable for each transmit buffer and transmit/receive FIFO buffer.<br>• Selects ID priority transmission or transmit buffer number priority transmission.<br>• Transmission can be aborted (possible to confirm with a flag)<br>• One-shot transmission function |
| Interval transmission function | Transmits messages at configurable intervals<br>(transmit mode or gateway mode of transmit/receive FIFO buffers) |
| Transmit queue function | Transmits all stored messages according to the ID priority. |
| Transmit history function | Stores the history information of transmission-completed messages<br>Adds timestamp (recording message transmission time as a 16-bit timer value) to the history information. |
| Gateway function | Transmits a received message automatically. |
| Bus off recovery mode selection | Selects the method for returning from bus off state.<br>• ISO11898-1 compliant<br>• Automatic entry to channel halt mode at bus-off entry<br>• Automatic entry to channel halt mode at bus-off end<br>• Transition to channel halt mode by program request<br>• Transition to the error-active state by program request (forcible return from the bus off state) |
| Error status monitoring | • Monitors CAN protocol errors (stuff error, form error, ACK error, CRC error, bit error, ACK delimiter error, and bus dominant lock).<br>• Detects error status transitions (error warning, error passive, bus off entry, and bus off recovery)<br>• Reads the error counter.<br>• Monitors DLC errors. |
| Interrupt factor | 20 factors<br>• Global Interrupts (2 factors)<br>Receive FIFO interrupt<br>Global error interrupt<br>• Channel interrupts (3 factors/channel)<br>CANm transmit interrupt (m = 0 to 5)<br>– CANm transmit complete interrupt<br>– CANm transmit abort interrupt<br>– CANm transmit/receive FIFO transmit complete interrupt (in transmit mode, gateway mode)<br>– CANm transmit history interrupt<br>– CANm transmit queue interrupt<br>CANm transmit/receive FIFO receive complete interrupt (in receive mode, gateway mode)<br>CANm error interrupt |
| CAN stop mode | Reduces power consumption by stopping clock supply to the RS-CANFD module. |

Table 21.12 RS-CANFD Module Specifications (3/3)

| Item | Specification |
|---|---|
| CAN clock source | Selects the clkc or the clk_xincan.<br>For the operating frequency range, see **Table 21.8**. |
| Test function | Test function for user evaluation<br>• Listen-only mode<br>• Self-test mode 0 (external loopback)<br>• Self-test mode 1 (internal loopback)<br>• Restricted operation mode<br>• RAM test (read/write test)<br>• Inter-channel communication test [CRC error test enabled] |

## 21.3.2 Interface Modes

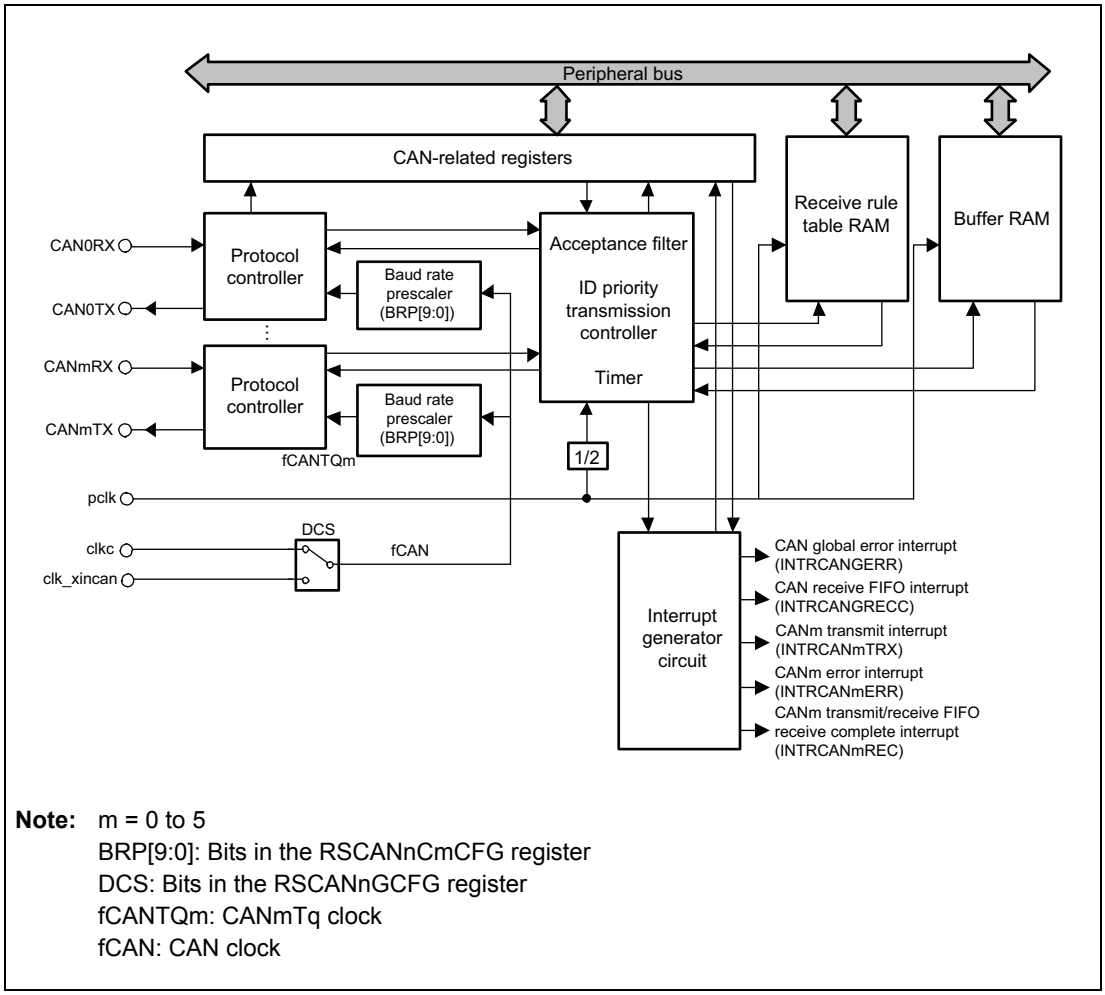
The RS-CANFD has two interface modes.

- Classical CAN mode: Handles only classical CAN frames.
- CAN FD mode: Handles classical CAN frames and CAN FD frames.

These two modes use different register maps with the same base address. Register maps change by switching interface modes.

Interface modes are switched by the RCMC bit in the RSCFDnCFDGRMCFG register.

### 21.3.3 Block Diagram




**Note:** m = 0 to 5
BRP[9:0]: Bits in the RSCANnCmCFG register
DCS: Bits in the RSCANnGCFG register
fCANTQm: CANmTq clock
fCAN: CAN clock

**Figure 21.1 RS-CANFD Module Block Diagram (Classical CAN Mode)**

In CAN FD mode, different clock signals are input to the baud rate prescaler and the protocol controller respectively. See **Section 21.11.1.3, Communication Speed Setting**.

# 21.4 Registers (Classical CAN Mode)

## 21.4.1 List of Registers

The following tables list RS-CANFD registers to be used in classical CAN mode.

For details about <RSCANn_base>, see **Section 21.2.1, Register Base Address**.

**Table 21.13 Registers (1/2)**

| Module | Register | Symbol | Address | Guard Group |
|---|---|---|---|---|
| **Channel-related registers** | | | | |
| RSCANn | Channel m Configuration register | RSCANnCmCFG | <RSCANn_base> + $0000_H$ + ($10_H$ × m) | RSCANn Chm |
| | Channel m control register | RSCANnCmCTR | <RSCANn_base> + $0004_H$ + ($10_H$ × m) | RSCANn Chm |
| | Channel m status register | RSCANnCmSTS | <RSCANn_base> + $0008_H$ + ($10_H$ × m) | RSCANn Chm |
| | Channel m error flag register | RSCANnCmERFL | <RSCANn_base> + $000C_H$ + ($10_H$ × m) | RSCANn Chm |
| **Global-related registers** | | | | |
| RSCANn | Global configuration register | RSCANnGCFG | <RSCANn_base> + $0084_H$ | RSCANn global |
| | Global control register | RSCANnGCTR | <RSCANn_base> + $0088_H$ | RSCANn global |
| | Global status register | RSCANnGSTS | <RSCANn_base> + $008C_H$ | RSCANn global |
| | Global error flag register | RSCANnGERFL | <RSCANn_base> + $0090_H$ | RSCANn global |
| | Global timestamp counter register | RSCANnGTSC | <RSCANn_base> + $0094_H$ | RSCANn global |
| | Global TX Interrupt Status register 0 | RSCANnGTINTSTS0 | <RSCANn_base> + $0460_H$ | RSCANn global |
| | Global TX Interrupt Status register 1 | RSCANnGTINTSTS1 | <RSCANn_base> + $0464_H$ | RSCANn global |
| | Global FD configuration register | RSCANnGFDCFG | <RSCANn_base> + $0474_H$ | RSCANn global |
| **Receive rule-related registers** | | | | |
| RSCANn | Receive Rule Entry Control register | RSCANnGAFLECTR | <RSCANn_base> + $0098_H$ | RSCANn global |
| | Receive Rule Configuration register 0 | RSCANnGAFLCFG0 | <RSCANn_base> + $009C_H$ | RSCANn global |
| | Receive Rule Configuration register 1 | RSCANnGAFLCFG1 | <RSCANn_base> + $00A0_H$ | RSCANn global |
| | Receive Rule ID register j | RSCANnGAFLIDj | <RSCANn_base> + $0500_H$ + ($10_H$ × j) | RSCANn global |
| | Receive Rule Mask register j | RSCANnGAFLMj | <RSCANn_base> + $0504_H$ + ($10_H$ × j) | RSCANn global |
| | Receive Rule Pointer 0 register j | RSCANnGAFLP0j | <RSCANn_base> + $0508_H$ + ($10_H$ × j) | RSCANn global |
| | Receive Rule Pointer 1 register j | RSCANnGAFLP1j | <RSCANn_base> + $050C_H$ + ($10_H$ × j) | RSCANn global |
| **Receive buffer-related registers** | | | | |
| RSCANn | Receive Buffer Number register | RSCANnRMNB | <RSCANn_base> + $00A4_H$ | RSCANn global |
| | Receive Buffer New Data register y | RSCANnRMNDy | <RSCANn_base> + $00A8_H$ + ($04_H$ × y) | RSCANn global |
| | Receive Buffer ID register q | RSCANnRMIDq | <RSCANn_base> + $0600_H$ + ($10_H$ × q) | RSCANn global |
| | Receive Buffer Pointer register q | RSCANnRMPTRq | <RSCANn_base> + $0604_H$ + ($10_H$ × q) | RSCANn global |
| | Receive Buffer Data Field 0 register q | RSCANnRMDF0q | <RSCANn_base> + $0608_H$ + ($10_H$ × q) | RSCANn global |
| | Receive Buffer Data Field 1 register q | RSCANnRMDF1q | <RSCANn_base> + $060C_H$ + ($10_H$ × q) | RSCANn global |
| **Receive FIFO buffer-related registers** | | | | |
| RSCANn | Receive FIFO Buffer Configuration and Control register x | RSCANnRFCCx | <RSCANn_base> + $00B8_H$ + ($04_H$ × x) | RSCANn global |
| | Receive FIFO Buffer Status register x | RSCANnRFSTSx | <RSCANn_base> + $00D8_H$ + ($04_H$ × x) | RSCANn global |
| | Receive FIFO Buffer Pointer Control register x | RSCANnRFPCTRx | <RSCANn_base> + $00F8_H$ + ($04_H$ × x) | RSCANn global |
| | Receive FIFO Buffer Access ID register x | RSCANnRFIDx | <RSCANn_base> + $0E00_H$ + ($10_H$ × x) | RSCANn global |
| | Receive FIFO Buffer Access Pointer register x | RSCANnRFPTRx | <RSCANn_base> + $0E04_H$ + ($10_H$ × x) | RSCANn global |
| | Receive FIFO Buffer Access Data Field 0 register x | RSCANnRFDF0x | <RSCANn_base> + $0E08_H$ + ($10_H$ × x) | RSCANn global |
| | Receive FIFO Buffer Access Data Field 1 register x | RSCANnRFDF1x | <RSCANn_base> + $0E0C_H$ + ($10_H$ × x) | RSCANn global |
| **Transmit/Receive FIFO buffer related registers** | | | | |

Table 21.13 Registers (2/2)

| Module | Register | Symbol | Address | Guard Group |
|---|---|---|---|---|
| RSCANn | Transmit/receive FIFO Buffer Configuration and Control register k | RSCANnCFCCk | <RSCANn_base> + $0118_H$ + ($04_H$ × k) | RSCANn global |
| | Transmit/receive FIFO Buffer Status register k | RSCANnCFSTSk | <RSCANn_base> + $0178_H$ + ($04_H$ × k) | RSCANn global |
| | Transmit/receive FIFO Buffer Pointer Control register k | RSCANnCFPCTRk | <RSCANn_base> + $01D8_H$ + ($04_H$ × k) | RSCANn global |
| | Transmit/receive FIFO Buffer Access ID register k | RSCANnCFIDk | <RSCANn_base> + $0E80_H$ + ($10_H$ × k) | RSCANn global |
| | Transmit/receive FIFO Buffer Access Pointer register k | RSCANnCFPTRk | <RSCANn_base> + $0E84_H$ + ($10_H$ × k) | RSCANn global |
| | Transmit/receive FIFO Buffer Access Data Field 0 register k | RSCANnCFDF0k | <RSCANn_base> + $0E88_H$ + ($10_H$ × k) | RSCANn global |
| | Transmit/receive FIFO Buffer Access Data Field 1 register k | RSCANnCFDF1k | <RSCANn_base> + $0E8C_H$ + ($10_H$ × k) | RSCANn global |
| **FIFO status-related registers** | | | | |
| RSCANn | FIFO Empty Status register | RSCANnFESTS | <RSCANn_base> + $0238_H$ | RSCANn global |
| | FIFO Full Status register | RSCANnFFSTS | <RSCANn_base> + $023C_H$ | RSCANn global |
| | FIFO Message Lost Status register | RSCANnFMSTS | <RSCANn_base> + $0240_H$ | RSCANn global |
| | Receive FIFO Buffer Interrupt Flag Status register | RSCANnRFISTS | <RSCANn_base> + $0244_H$ | RSCANn global |
| | Transmit/receive FIFO Buffer Receive Interrupt Flag Status register | RSCANnCFRISTS | <RSCANn_base> + $0248_H$ | RSCANn global |
| | Transmit/receive FIFO Buffer Transmit Interrupt Flag Status register | RSCANnCFTISTS | <RSCANn_base> + $024C_H$ | RSCANn global |
| **Transmit buffer-related registers** | | | | |
| RSCANn | Transmit Buffer Control register p | RSCANnTMCp | <RSCANn_base> + $0250_H$ + ($01_H$ × p) | RSCANn global |
| | Transmit Buffer Status register p | RSCANnTMSTSp | <RSCANn_base> + $02D0_H$ + ($01_H$ × p) | RSCANn global |
| | Transmit Buffer ID register p | RSCANnTMIDp | <RSCANn_base> + $1000_H$ + ($10_H$ × p) | RSCANn global |
| | Transmit Buffer Pointer register p | RSCANnTMPTRp | <RSCANn_base> + $1004_H$ + ($10_H$ × p) | RSCANn global |
| | Transmit Buffer Data Field 0 register p | RSCANnTMDF0p | <RSCANn_base> + $1008_H$ + ($10_H$ × p) | RSCANn global |
| | Transmit Buffer Data Field 1 register p | RSCANnTMDF1p | <RSCANn_base> + $100C_H$ + ($10_H$ × p) | RSCANn global |
| | Transmit Buffer Interrupt Enable Configuration register y | RSCANnTMIECy | <RSCANn_base> + $0390_H$ + ($04_H$ × y) | RSCANn global |
| **Transmit buffer status-related registers** | | | | |
| RSCANn | Transmit Buffer Transmit Request Status register y | RSCANnTMTRSTSy | <RSCANn_base> + $0350_H$ + ($04_H$ × y) | RSCANn global |
| | Transmit Buffer Transmit Abort Request Status register y | RSCANnTMTARSTSy | <RSCANn_base> + $0360_H$ + ($04_H$ × y) | RSCANn global |
| | Transmit Buffer Transmit Complete Status register y | RSCANnTMTCSTSy | <RSCANn_base> + $0370_H$ + ($04_H$ × y) | RSCANn global |
| | Transmit Buffer Transmit Abort Status register y | RSCANnTMTASTSy | <RSCANn_base> + $0380_H$ + ($04_H$ × y) | RSCANn global |
| **Transmit queue-related registers** | | | | |
| RSCANn | Transmit Queue Configuration and Control register m | RSCANnTXQCCm | <RSCANn_base> + $03A0_H$ + ($04_H$ × m) | RSCANn Chm |
| | Transmit Queue Status register m | RSCANnTXQSTSm | <RSCANn_base> + $03C0_H$ + ($04_H$ × m) | RSCANn Chm |
| | Transmit Queue Pointer Control register m | RSCANnTXQPCTRm | <RSCANn_base> + $03E0_H$ + ($04_H$ × m) | RSCANn Chm |
| **Transmit history-related registers** | | | | |
| RSCANn | Transmit History Configuration and Control register m | RSCANnTHLCCm | <RSCANn_base> + $0400_H$ + ($04_H$ × m) | RSCANn Chm |
| | Transmit History Status register m | RSCANnTHLSTSm | <RSCANn_base> + $0420_H$ + ($04_H$ × m) | RSCANn Chm |
| | Transmit History Pointer Control register m | RSCANnTHLPCTRm | <RSCANn_base> + $0440_H$ + ($04_H$ × m) | RSCANn Chm |
| | Transmit History Access register m | RSCANnTHLACCm | <RSCANn_base> + $1800_H$ + ($04_H$ × m) | RSCANn Chm |
| **Test-related registers** | | | | |
| RSCANn | Global Test Configuration register | RSCANnGTSTCFG | <RSCANn_base> + $0468_H$ | RSCANn global |
| RSCANn | Global Test Control register | RSCANnGTSTCTR | <RSCANn_base> + $046C_H$ | RSCANn global |
| | Global Lock Key register | RSCANnGLOCKK | <RSCANn_base> + $047C_H$ | RSCANn global |
| | RAM Test Page Access register r | RSCANnRPGACCr | <RSCANn_base> + $1900_H$ + ($04_H$ × r) | RSCANn global |

**Note:** For details about Guard Group, see **Section 33, Functional Safety**.

**Table 21.14 Transmit Buffer p Allocated to Each Channel**

| | CANm |
|---|---|
| Transmit buffer p | Transmit buffer 16 × m + 0 |
| | Transmit buffer 16 × m + 1 |
| | Transmit buffer 16 × m + 2 |
| | Transmit buffer 16 × m + 3 |
| | Transmit buffer 16 × m + 4 |
| | Transmit buffer 16 × m + 5 |
| | Transmit buffer 16 × m + 6 |
| | Transmit buffer 16 × m + 7 |
| | Transmit buffer 16 × m + 8 |
| | Transmit buffer 16 × m + 9 |
| | Transmit buffer 16 × m + 10 |
| | Transmit buffer 16 × m + 11 |
| | Transmit buffer 16 × m + 12 |
| | Transmit buffer 16 × m + 13 |
| | Transmit buffer 16 × m + 14 |
| | Transmit buffer 16 × m + 15 |

**Table 21.15 Transmit/Receive FIFO Buffer k Allocated to Each Channel**

| | CANm |
|---|---|
| Transmit/receive FIFO buffer k | Transmit/receive FIFO buffer 3 × m + 0 |
| | Transmit/receive FIFO buffer 3 × m + 1 |
| | Transmit/receive FIFO buffer 3 × m + 2 |

**Table 21.16 Transmit Buffer p Linked to the Transmit/Receive FIFO Buffer by the Setting of Bits CFTML[3:0]**

| Setting of Bits CFTML[3:0] | Transmit Buffer p Linked to the Transmit/Receive FIFO Buffer |
|---|---|
| $0000_B$ | Transmit buffer 16 × m + 0 |
| $0001_B$ | Transmit buffer 16 × m + 1 |
| $0010_B$ | Transmit buffer 16 × m + 2 |
| $0011_B$ | Transmit buffer 16 × m + 3 |
| $0100_B$ | Transmit buffer 16 × m + 4 |
| $0101_B$ | Transmit buffer 16 × m + 5 |
| $0110_B$ | Transmit buffer 16 × m + 6 |
| $0111_B$ | Transmit buffer 16 × m + 7 |
| $1000_B$ | Transmit buffer 16 × m + 8 |
| $1001_B$ | Transmit buffer 16 × m + 9 |
| $1010_B$ | Transmit buffer 16 × m + 10 |
| $1011_B$ | Transmit buffer 16 × m + 11 |
| $1100_B$ | Transmit buffer 16 × m + 12 |
| $1101_B$ | Transmit buffer 16 × m + 13 |
| $1110_B$ | Transmit buffer 16 × m + 14 |
| $1111_B$ | Transmit buffer 16 × m + 15 |

**Table 21.17 Transmit Buffer p Allocated to the Transmit Queue of Each Channel**

| Setting of Bits TXQDC[3:0] | Transmit Buffer p Allocated to the Transmit Queue |
|---|---|
| $0000_B$ | Setting prohibited |
| $0001_B$ | Setting prohibited |
| $0010_B$ | Transmit buffer 16 × m + 15 to 16 × m + 13 |
| $0011_B$ | Transmit buffer 16 × m + 15 to 16 × m + 12 |
| $0100_B$ | Transmit buffer 16 × m + 15 to 16 × m + 11 |
| $0101_B$ | Transmit buffer 16 × m + 15 to 16 × m + 10 |
| $0110_B$ | Transmit buffer 16 × m + 15 to 16 × m + 9 |
| $0111_B$ | Transmit buffer 16 × m + 15 to 16 × m + 8 |
| $1000_B$ | Transmit buffer 16 × m + 15 to 16 × m + 7 |
| $1001_B$ | Transmit buffer 16 × m + 15 to 16 × m + 6 |
| $1010_B$ | Transmit buffer 16 × m + 15 to 16 × m + 5 |
| $1011_B$ | Transmit buffer 16 × m + 15 to 16 × m + 4 |
| $1100_B$ | Transmit buffer 16 × m + 15 to 16 × m + 3 |
| $1101_B$ | Transmit buffer 16 × m + 15 to 16 × m + 2 |
| $1110_B$ | Transmit buffer 16 × m + 15 to 16 × m + 1 |
| $1111_B$ | Transmit buffer 16 × m + 15 to 16 × m + 0 |

## 21.4.2 Details of Channel-Related Registers

### 21.4.2.1 RSCANnCmCFG — Channel Configuration Register (m = 0 to 5)

**Access:** RSCANnCmCFG register can be read or written in 32-bit units
RSCANnCmCFGL, RSCANnCmCFGH registers can be read or written in 16-bit units
RSCANnCmCFGLL, RSCANnCmCFGLH, RSCANnCmCFGHL, RSCANnCmCFGHH registers can be read or written in 8-bit units

**Address:** RSCANnCmCFG: <RSCANn_base> + $0000_H$ + ($10_H$ × m)

RSCANnCmCFGL: <RSCANn_base> + $0000_H$ + ($10_H$ × m),
RSCANnCmCFGH: <RSCANn_base> + $0002_H$ + ($10_H$ × m)

RSCANnCmCFGLL: <RSCANn_base> + $0000_H$ + ($10_H$ × m),
RSCANnCmCFGLH: <RSCANn_base> + $0001_H$ + ($10_H$ × m),
RSCANnCmCFGHL: <RSCANn_base> + $0002_H$ + ($10_H$ × m),
RSCANnCmCFGHH: <RSCANn_base> + $0003_H$ + ($10_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | SJW[1:0] | | — | TSEG2[2:0] | | | TSEG1[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | BRP[9:0] | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.18 RSCANnCmCFG Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 25, 24 | SJW[1:0] | Resynchronization Jump Width Control<br>b25 b24<br>0 0: 1 Tq<br>0 1: 2 Tq<br>1 0: 3 Tq<br>1 1: 4 Tq |
| 23 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 22 to 20 | TSEG2[2:0] | Time Segment 2 Control<br>b22 b21 b20<br>0 0 0: Setting prohibited<br>0 0 1: 2 Tq<br>0 1 0: 3 Tq<br>0 1 1: 4 Tq<br>1 0 0: 5 Tq<br>1 0 1: 6 Tq<br>1 1 0: 7 Tq<br>1 1 1: 8 Tq |

Table 21.18 RSCANnCmCFG Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 19 to 16 | TSEG1[3:0] | Time Segment 1 Control<br>b19 b18 b17 b16<br>0 0 0 0: Setting prohibited<br>0 0 0 1: Setting prohibited<br>0 0 1 0: Setting prohibited<br>0 0 1 1: 4 Tq<br>0 1 0 0: 5 Tq<br>0 1 0 1: 6 Tq<br>0 1 1 0: 7 Tq<br>0 1 1 1: 8 Tq<br>1 0 0 0: 9 Tq<br>1 0 0 1: 10 Tq<br>1 0 1 0: 11 Tq<br>1 0 1 1: 12 Tq<br>1 1 0 0: 13 Tq<br>1 1 0 1: 14 Tq<br>1 1 1 0: 15 Tq<br>1 1 1 1: 16 Tq |
| 15 to 10 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 9 to 0 | BRP[9:0] | Prescaler Division Ratio Set<br>When these bits are set to P (0 to 1023), the baud rate prescaler divides fCAN by P + 1. |

Modify the RSCANnCmCFG register in channel reset mode or channel halt mode. Set this register in channel reset mode before transitioning to channel communication mode or channel halt mode. For a description of the bit timing parameters and settings, see **Section 21.11.1, Initial Settings**.

#### SJW[1:0] Bits

These bits are used to specify a Tq value for the resynchronization jump width. Allowed values are 1 Tq to 4 Tq, inclusive.
Set a value less than or equal to the value of the TSEG2 bits.

#### TSEG2[2:0] Bits

These bits are used to specify a Tq value for the length of phase segment 2 (PHASE_SEG2).
Allowed values are 2 Tq to 8 Tq, inclusive.
Set a value smaller than the value of the TSEG1 bits.

#### TSEG1[3:0] Bits

These bits are used to specify a Tq value for the total length of the propagation segment (PROP_SEG) and phase segment 1 (PHASE_SEG1).
Allowed values are 4 Tq to 16 Tq, inclusive.

#### BRP[9:0] Bits

The CANmTq clock (fCANTQm) is calculated by dividing the CAN clock (fCAN) by the baud rate prescaler, ((BRP[9:0]) + 1). One clock cycle of the CANmTq clock is 1 Time Quantum (Tq).

### 21.4.2.2 RSCANnCmCTR — Channel Control Register (m = 0 to 5)

**Access:** RSCANnCmCTR register can be read or written in 32-bit units
RSCANnCmCTRL, RSCANnCmCTRH registers can be read or written in 16-bit units
RSCANnCmCTRLL, RSCANnCmCTRLH, RSCANnCmCTRHL, RSCANnCmCTRHH registers can be read or written in 8-bit units

**Address:** RSCANnCmCTR: <RSCANn_base> + $0004_H$ + ($10_H$ × m)

RSCANnCmCTRL: <RSCANn_base> + $0004_H$ + ($10_H$ × m),
RSCANnCmCTRH: <RSCANn_base> + $0006_H$ + ($10_H$ × m)

RSCANnCmCTRLL: <RSCANn_base> + $0004_H$ + ($10_H$ × m),
RSCANnCmCTRLH: <RSCANn_base> + $0005_H$ + ($10_H$ × m),
RSCANnCmCTRHL: <RSCANn_base> + $0006_H$ + ($10_H$ × m),
RSCANnCmCTRHH: <RSCANn_base> + $0007_H$ + ($10_H$ × m)

**Value after reset:** $0000\ 0005_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | CRCT | — | — | — | CTMS[1:0] | | CTME | ERRD | BOM[1:0] | | — | — | — | — | TAIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ALIE | BLIE | OLIE | BORIE | BOEIE | EPIE | EWIE | BEIE | — | — | — | — | RTBO | CSLPR | CHMDC[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 21.19 RSCANnCmCTR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 30 | CRCT | CRC Error Test Enable<br>0: The first bit of the reception ID field is not inverted.<br>1: The first bit of the reception ID field is inverted. |
| 29 to 27 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 26, 25 | CTMS[1:0] | Communication Test Mode Select<br>b26 b25<br>0 0: Standard test mode<br>0 1: Listen-only mode<br>1 0: Self-test mode 0 (external loopback mode)<br>1 1: Self-test mode 1 (internal loopback mode) |
| 24 | CTME | Communication Test Mode Enable<br>0: Communication test mode is disabled.<br>1: Communication test mode is enabled. |
| 23 | ERRD | Error Display Mode Select<br>0: Error flags are displayed only for the first error information after bits 14 to 8 in the RSCANnCmERFL register are all cleared.<br>1: Error flags for all error information are displayed. |
| 22, 21 | BOM[1:0] | Bus Off Recovery Mode Select<br>b22 b21<br>0 0: ISO11898-1 compliant<br>0 1: Entry to channel halt mode automatically at bus-off entry<br>1 0: Entry to channel halt mode automatically at bus-off end<br>1 1: Entry to channel halt mode (in bus-off state) by program request |
| 20 to 17 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 16 | TAIE | Transmit Abort Interrupt Enable<br>0: Transmit abort interrupt is disabled.<br>1: Transmit abort interrupt is enabled. |

Table 21.19 RSCANnCmCTR Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | ALIE | Arbitration Lost Interrupt Enable<br>0: Arbitration lost interrupt is disabled.<br>1: Arbitration lost interrupt is enabled. |
| 14 | BLIE | Bus Lock Interrupt Enable<br>0: Bus lock interrupt is disabled.<br>1: Bus lock interrupt is enabled. |
| 13 | OLIE | Overload Frame Transmit Interrupt Enable<br>0: Overload frame transmit interrupt is disabled.<br>1: Overload frame transmit interrupt is enabled. |
| 12 | BORIE | Bus Off Recovery Interrupt Enable<br>0: Bus off recovery interrupt is disabled.<br>1: Bus off recovery interrupt is enabled. |
| 11 | BOEIE | Bus Off Entry Interrupt Enable<br>0: Bus off entry interrupt is disabled.<br>1: Bus off entry interrupt is enabled. |
| 10 | EPIE | Error Passive Interrupt Enable<br>0: Error passive interrupt is disabled.<br>1: Error passive interrupt is enabled. |
| 9 | EWIE | Error Warning Interrupt Enable<br>0: Error warning interrupt is disabled.<br>1: Error warning interrupt is enabled. |
| 8 | BEIE | Bus Error Interrupt Enable<br>0: Bus error interrupt is disabled.<br>1: Bus error interrupt is enabled. |
| 7 to 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | RTBO | Forcible Return from Bus-off<br>When this bit is set to 1, forcible return from the bus off state is made. This bit is always read as 0. |
| 2 | CSLPR | Channel Stop Mode<br>0: Other than channel stop mode<br>1: Channel stop mode |
| 1, 0 | CHMDC[1:0] | Mode Select<br>b1 b0<br>0 0: Channel communication mode<br>0 1: Channel reset mode<br>1 0: Channel halt mode<br>1 1: Setting prohibited |

### CRCT Bit

This bit is used to test the CRC generation circuit in the RS-CANFD module. Setting this bit to 1 inverts the first bit of the ID field when a message is received. With this inversion of bit, the CRC calculation result does not match the normal CRC value of the received frame, which can detect a CRC error (the CERR bit in the RSCANnCmERFL register is 1). When using this function, note the following.

- This function is available while the CTME bit in the RSCANnCmCTR register is 1 (communication test mode enabled).
- This function cannot communicate with other CAN nodes. Use this function for inter-channel communication test (the CmICBCE bit in the RSCANnGTSTCFG register is 1).
- Bit inversion in the ID field may cause bit stuffing rule violation. In that case, no CRC error is detected but a stuff error is detected.

Modify this bit only in channel halt mode. This bit is always 0 in channel reset mode.

#### CTMS[1:0] Bits

These bits are used to select a communication test mode. Modify these bits in channel halt mode only. These bits are set to 0 in channel reset mode.

#### CTME Bit

Setting this bit to 1 enables communication test mode. Modify this bit in channel halt mode. This bit is set to 0 in channel reset mode.

#### ERRD Bit

This bit is used to control the display mode of bits 14 to 8 in the RSCANnCmERFL register.
When this bit is clear to 0, if any error is detected while the flags of bits 14 to 8 in the RSCANnCmERFL register are all 0, only the flags of the first error event are set to 1. If two or more errors occur in the first error event, all the flags of the detected errors are set to 1.

When this bit is set to 1, all the flags of errors that have occurred are set to 1 regardless of the error occurrence order.

Modify this bit only in channel reset mode or channel halt mode.

#### BOM[1:0] Bits

These bits are used to select the bus off recovery mode of the RS-CANFD module.

When the BOM[1:0] bits are set to $00_B$, return from the bus off state to the error active state is compliant with the CAN specifications. That is, the RS-CANFD module reenters the CAN communication (error active state) after 11 consecutive recessive bits are detected 128 times. A bus off recovery interrupt request is generated at the time of return from the bus off state. Even if the CHMDC[1:0] bits are set to $10_B$ (channel halt mode) before recessive bits are detected 128 times, the RS-CANFD module does not transition to channel halt mode until recessive bits are detected 128 times.

When the RS-CANFD module reaches the bus off state when the BOM[1:0] bits are set to $01_B$, the CHMDC[1:0] bits in the RSCANnCmCTR register (m = 0 to 5) are set to $10_B$ and the RS-CANFD module transitions to channel halt mode. No bus off recovery interrupt request is generated and the TEC[7:0] and REC[7:0] bits in the RSCANnCmSTS register are cleared to $00_H$.

When the RS-CANFD module reaches the bus off state when the BOM[1:0] bits are set to $10_B$, the CHMDC[1:0] bits are set to $10_B$ and the RS-CANFD module transitions to channel halt mode after return from the bus off state (11 consecutive recessive bits are detected 128 times). A bus off recovery interrupt request is generated at the time of return from the bus off state and the TEC[7:0] and REC[7:0] bits are cleared to $00_H$.

When the BOM[1:0] bits are set to $11_B$ and the CHMDC[1:0] bits are set to $10_B$ while the RS-CANFD module is in the bus off state, the RS-CANFD module transitions to channel halt mode. No bus off recovery interrupt request is generated at the time of return from the bus off state and the TEC[7:0] and REC[7:0] bits are cleared to $00_H$. However, if 11 consecutive recessive bits are detected 128 times and the RS-CANFD module has recovered to the error active state from the bus off state before the CHMDC[1:0] bits are set to $10_B$, a bus off recovery interrupt request is generated.

If a program writes to the CHMDC[1:0] bits at the same time as the RS-CANFD module transition to channel halt mode (at bus off entry when the BOM[1:0] bits are $01_B$ or at bus off end when the BOM[1:0] bits are $10_B$), the program's writing takes precedence. Modify the BOM[1:0] bits only in channel reset mode.

#### TAIE Bit

When transmit abort of the transmit buffer is completed with the TAIE bit set to 1, an interrupt request is generated. Modify this bit only in channel reset mode.

#### ALIE Bit

When the ALF flag in the RSCANnCmERFL register is set to 1 with the ALIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### BLIE Bit

When the BLF flag in the RSCANnCmERFL register is set to 1 with the BLIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### OLIE Bit

When the OVLF flag in the RSCANnCmERFL register is set to 1 with the OLIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### BORIE Bit

When the BORF flag in the RSCANnCmERFL register is set to 1 with the BORIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### BOEIE Bit

When the BOEF flag in the RSCANnCmERFL register is set to 1 with the BOEIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### EPIE Bit

When the EPF flag in the RSCANnCmERFL register is set to 1 with the EPIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### EWIE Bit

When the EWF flag in the RSCANnCmERFL register is set to 1 with the EWIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### BEIE Bit

When the BEF flag in the RSCANnCmERFL register is set to 1 with the BEIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### RTBO Bit

Setting this bit to 1 in the bus off state forcibly returns the state from the bus off state to the error active state. This bit is automatically cleared to 0. Setting this bit to 1 clears the TEC[7:0] and REC[7:0] bits in the RSCANnCmSTS register to $00_H$ and also clears the BOSTS flag in the RSCANnCmSTS register to 0 (not in bus off state). The other registers remain unchanged. No bus off recovery interrupt request is generated upon return from the bus off state in this case. Use this bit only when the BOM[1:0] bits in the RSCANnCmCTR register are $00_B$ (ISO11898-1 compliant).

A delay of up to 1 CAN bit time occurs after the RTBO bit is set to 1 until the RS-CANFD module transitions to the error active state. Set this bit to 1 in channel communication mode.

#### CSLPR Bit

Setting this bit to 1 places the channel into channel stop mode.
Clearing this bit to 0 makes the channel exit channel stop mode.
This bit should not be modified in channel communication mode or channel halt mode.

#### CHMDC[1:0] Bits

These bits are used to select a channel mode (channel communication mode, channel reset mode, or channel halt mode). For details, see **Section 21.7.2, Channel Modes**. Setting the CSLPR bit to 1 in channel reset mode allows transition to channel stop mode. Do not set the CHMDC[1:0] bits to $11_B$. When the RS-CANFD module has automatically transitioned to channel halt mode based on the setting of the BOM[1:0] bits, the CHMDC[1:0] bits automatically become $10_B$.

### 21.4.2.3 RSCANnCmSTS — Channel Status Register (m = 0 to 5)

**Access:** RSCANnCmSTS register is a read-only register that can be read in 32-bit units
RSCANnCmSTSL, RSCANnCmSTSH registers are read-only registers that can be read in 16-bit units
RSCANnCmSTSLL, RSCANnCmSTSHL, RSCANnCmSTSHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnCmSTS: <RSCANn_base> + $0008_H$ + ($10_H$ × m)

RSCANnCmSTSL: <RSCANn_base> + $0008_H$ + ($10_H$ × m),
RSCANnCmSTSH: <RSCANn_base> + $000A_H$ + ($10_H$ × m)

RSCANnCmSTSLL: <RSCANn_base> + $0008_H$ + ($10_H$ × m),
RSCANnCmSTSHL: <RSCANn_base> + $000A_H$ + ($10_H$ × m),
RSCANnCmSTSHH: <RSCANn_base> + $000B_H$ + ($10_H$ × m)

**Value after reset:** $0000\ 0005_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEC[7:0] | | | | | | | | REC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | COMSTS | RECSTS | TRMSTS | BOSTS | EPSTS | CSLPSTS | CHLTSTS | CRSTSTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.20 RSCANnCmSTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | TEC[7:0] | The transmit error counter (TEC) can be read. |
| 23 to 16 | REC[7:0] | The receive error counter (REC) can be read. |
| 15 to 8 | Reserved | When read, the value after reset is returned. |
| 7 | COMSTS | Communication Status Flag<br>0: Communication is not ready.<br>1: Communication is ready. |
| 6 | RECSTS | Receive Status Flag<br>0: Bus idle, in transmission or bus off state<br>1: In reception |
| 5 | TRMSTS | Transmit Status Flag<br>0: Bus idle or in reception<br>1: In transmission or bus off state |
| 4 | BOSTS | Bus Off Status Flag<br>0: Not in bus off state<br>1: In bus off state |
| 3 | EPSTS | Error Passive Status Flag<br>0: Not in error passive state<br>1: In error passive state |
| 2 | CSLPSTS | Channel Stop Status Flag<br>0: Not in channel stop mode<br>1: In channel stop mode |
| 1 | CHLTSTS | Channel Halt Status Flag<br>0: Not in channel halt mode<br>1: In channel halt mode |
| 0 | CRSTSTS | Channel Reset Status Flag<br>0: Not in channel reset mode<br>1: In channel reset mode |

#### TEC[7:0] Bits

These bits contain the transmit error counter value. For transmit error counter increment/decrement conditions, see the CAN specification (ISO11898-1).

These bits are cleared to 0 in channel reset mode.

#### REC[7:0] Bits

These bits contain the receive error counter value. For receive error counter increment/decrement conditions, see the CAN specifications (ISO11898-1).

These bits are cleared to 0 in channel reset mode.

#### COMSTS Flag

This bit indicates that communication is ready.

This flag becomes 1 when the RS-CANFD module has detected 11 consecutive recessive bits after it has transitioned from channel reset mode or channel halt mode to channel communication mode. This flag is cleared to 0 in channel reset mode or channel halt mode.

#### RECSTS Flag

This flag is set to 1 when reception has started, and is cleared to 0 when the bus has become idle or transmission has started.

#### TRMSTS Flag

This flag is set to 1 when transmission has started, and is cleared to 0 when the bus has become idle or reception has started. This flag remains 1 in the bus off state.

#### BOSTS Flag

This flag is set to 1 when the bus off state (TEC[7:0] > 255) is entered. It is cleared to 0 when the RS-CANFD module has exited the bus off state.

#### EPSTS Flag

This flag is set to 1 when the RS-CANFD module has entered the error passive state ((128 ≤ TEC[7:0] ≤ 255) or (128 ≤ REC[7:0])). It is cleared to 0 when the RS-CANFD module has exited the error passive state or has entered channel reset mode.

#### CSLPSTS Flag

This flag is set to 1 when the RS-CANFD module has transitioned to channel stop mode, and is cleared to 0 when the RS-CANFD module has returned from channel stop mode.

#### CHLTSTS Flag

This flag is set to 1 when the RS-CANFD module has transitioned to channel halt mode, and is cleared to 0 when the RS-CANFD module has returned from channel halt mode.

**CRSTSTS Flag**

This flag is set to 1 when the RS-CANFD module has transitioned to channel reset mode, and is cleared to 0 when the RS-CANFD module has transitioned to channel communication mode or channel halt mode. This flag remains 1 when the RS-CANFD module transitions from channel reset mode to channel stop mode.

### 21.4.2.4 RSCANnCmERFL — Channel Error Flag Register (m = 0 to 5)

**Access:** RSCANnCmERFL register can be read or written in 32-bit units
RSCANnCmERFLL register can be read or written in 16-bit units
RSCANnCmERFLH register is a read-only register that can be read in 16-bit units
RSCANnCmERFLLL, RSCANnCmERFLLH registers can be read or written in 8-bit units
RSCANnCmERFLHL, RSCANnCmERFLHH registers are the read-only registers that can be read in 8-bit units

**Address:** RSCANnCmERFL: <RSCANn_base> + $000C_H$ + ($10_H$ × m)

RSCANnCmERFLL: <RSCANn_base> + $000C_H$ + ($10_H$ × m),
RSCANnCmERFLH: <RSCANn_base> + $000E_H$ + ($10_H$ × m)

RSCANnCmERFLLL: <RSCANn_base> + $000C_H$ + ($10_H$ × m),
RSCANnCmERFLLH: <RSCANn_base> + $000D_H$ + ($10_H$ × m),
RSCANnCmERFLHL: <RSCANn_base> + $000E_H$ + ($10_H$ × m),
RSCANnCmERFLHH: <RSCANn_base> + $000F_H$ + ($10_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | CRCREG[14:0] | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | ADERR | B0ERR | B1ERR | CERR | AERR | FERR | SERR | ALF | BLF | OVLF | BORF | BOEF | EPF | EWF | BEF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.21 RSCANnCmERFL Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 30 to 16 | CRCREG[14:0] | CRC Calculation Data<br>A CRC value calculated based on the transmit message or receive message is indicated. |
| 15 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 14 | ADERR | ACK Delimiter Error Flag<br>0: No ACK delimiter error is detected.<br>1: ACK delimiter error is detected. |
| 13 | B0ERR | Dominant Bit Error Flag<br>0: No dominant bit error is detected.<br>1: Dominant bit error is detected. |
| 12 | B1ERR | Recessive Bit Error Flag<br>0: No recessive bit error is detected.<br>1: Recessive bit error is detected. |
| 11 | CERR | CRC Error Flag<br>0: No CRC error is detected.<br>1: CRC error is detected. |
| 10 | AERR | ACK Error Flag<br>0: No ACK error is detected.<br>1: ACK error is detected. |
| 9 | FERR | Form Error Flag<br>0: No form error is detected.<br>1: Form error is detected. |

Table 21.21 RSCANnCmERFL Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 8 | SERR | Stuff Error Flag<br>0: No stuff error is detected.<br>1: Stuff error is detected. |
| 7 | ALF | Arbitration-lost Flag<br>0: No arbitration-lost is detected.<br>1: Arbitration-lost is detected. |
| 6 | BLF | Bus Lock Flag<br>0: No channel bus lock is detected.<br>1: Channel bus lock is detected. |
| 5 | OVLF | Overload Flag<br>0: No overload is detected.<br>1: Overload is detected. |
| 4 | BORF | Bus Off Recovery Flag<br>0: No bus off recovery is detected.<br>1: Bus off recovery is detected. |
| 3 | BOEF | Bus Off Entry Flag<br>0: No bus off entry is detected.<br>1: Bus off entry is detected. |
| 2 | EPF | Error Passive Flag<br>0: No error passive is detected.<br>1: Error passive is detected. |
| 1 | EWF | Error Warning Flag<br>0: No error warning is detected.<br>1: Error warning is detected. |
| 0 | BEF | Bus Error Flag<br>0: No channel bus error is detected.<br>1: Channel bus error is detected. |

See the CAN specification (ISO11898-1) for a description of error occurrence conditions. To clear each flag of this register, the program must write a 0 to the corresponding bit. These flags cannot be set to 1 by the program. If any of these flags is set to 1 at the same time that the program writes 0 to the flag, the flag is still set to 1. The channel reset mode transition clears all of these flags to 0.

If the ERRD bit in the RSCANnCmCTR register is set to 0 (ie, only the flags for the first error event are displayed) and an error related to bits 14 to 8 of RSCANnCmERFL is detected, the flag bits are only set to 1 by the error event if bits 14 to 8 were all 0 at the time when the error occurred.

### CRCREG[14:0] Flag

When the CTME bit in the RSCANnCmCTR register is set to 1 (communication test mode is enabled), the CRC value calculated based on the transmit or receive message can be read. When the CTME bit is set to 0 (communication test mode is disabled), these bits are always read as 0.

### ADERR Flag

This flag is set to 1 when a form error has been detected in the ACK delimiter during transmission.

### B0ERR Flag

This flag is set to 1 when a recessive bit has been detected though a dominant bit was transmitted.

### B1ERR Flag

This flag is set to 1 when a dominant bit has been detected though a recessive bit was transmitted.

#### CERR Flag

This flag is set to 1 when a CRC error has been detected.

#### AERR Flag

This flag is set to 1 when an ACK error has been detected.

#### FERR Flag

This flag is set to 1 when a form error has been detected.

#### SERR Flag

This flag is set to 1 when a stuff error has been detected.

#### ALF Flag

This flag is set to 1 when an arbitration-lost has been detected.

#### BLF Flag

This flag is set to 1 when 32 consecutive dominant bits have been detected on the CAN bus in channel communication mode. After that, detection of a dominant lock is restarted when either of the following conditions is met.

- A recessive bit is detected after the BLF bit has been cleared from 1 to 0.
- The RS-CANFD module transitions to channel reset mode and returns to channel communication mode after the BLF bit has been cleared from 1 to 0.

#### OVLF Flag

This flag is set to 1 when the overload frame transmit condition has been detected when performing reception or transmission.

#### BORF Flag

This flag is set to 1 when 11 consecutive recessive bits have been detected 128 times and the RS-CANFD module returns from the bus off state. However, this flag is not set to 1 if the RS-CANFD module returns from the bus off state in any of the following ways before 11 consecutive recessive bits are detected 128 times.

- The CHMDC[1:0] bits in the RSCANnCmCTR register are set to $01_B$ (channel reset mode).
- The RTBO bit in the RSCANnCmCTR register is set to 1 (forcible return from the bus off state is made).
- The BOM[1:0] bits in the RSCANnCmCTR register are set to $01_B$ (transition to channel halt mode at bus off entry).
- The CHMDC[1:0] bits in the RSCANnCmCTR register are set to $10_B$ (channel halt mode) before 11 consecutive recessive bits are detected 128 times with the BOM[1:0] bits set to $11_B$ (transition to channel halt mode upon a request from the program during bus off).

### BOEF Flag

This flag is set to 1 when the bus off state is reached (TEC[7:0] value > 255). This flag is also set to 1 if the bus off state is reached when the BOM[1:0] bits in the RSCANnCmCTR register (m = 0 to 5) are set to $01_B$ (transition to channel halt mode at bus off entry).

### EPF Flag

This flag becomes 1 when the error passive state is reached (REC[7:0] or TEC[7:0] value > 127). This flag becomes 1 only when the REC[7:0] or TEC[7:0] value first exceeds 127. Therefore, if the program writes 0 to this flag while the value of REC[7:0] or TEC[7:0] remains over 127, this bit is not set to 1 until both REC [7:0] and TEC[7:0] values become 127 or less and then the REC[7:0] or TEC[7:0] value exceeds 127 again.

### EWF Flag

This flag is set to 1 only when the REC[7:0] or TEC[7:0] value first exceeds 95. Therefore, if the program writes 0 to this flag while the value of REC[7:0] or TEC[7:0] remains over 95, this bit is not set to 1 until both REC [7:0] and TEC[7:0] values become 95 or less and then the REC[7:0] or TEC[7:0] value exceeds 95 again.

### BEF Flag

This flag is set to 1 when any one of the ADERR, B0ERR, B1ERR, CERR, AERR, FERR, and SERR flags in the RSCANnCmERFL register is set to 1.

**NOTE**

To clear the flag of this register to 0, use a store instruction to write “0” to the given flag and “1” to the other flags.

## 21.4.3 Details of Global-Related Registers

### 21.4.3.1 RSCANnGCFG — Global Configuration Register

**Access:** RSCANnGCFG register can be read or written in 32-bit units
RSCANnGCFGL, RSCANnGCFGH registers can be read or written in 16-bit units
RSCANnGCFGLL, RSCANnGCFGLH, RSCANnGCFGHL, RSCANnGCFGHH registers can be read or written in 8-bit units

**Address:** RSCANnGCFG: <RSCANn_base> + $0084_H$

RSCANnGCFGL: <RSCANn_base> + $0084_H$,
RSCANnGCFGH: <RSCANn_base> + $0086_H$

RSCANnGCFGLL: <RSCANn_base> + $0084_H$,
RSCANnGCFGLH: <RSCANn_base> + $0085_H$,
RSCANnGCFGHL: <RSCANn_base> + $0086_H$,
RSCANnGCFGHH: <RSCANn_base> + $0087_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ITRCP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSBTCS[2:0] | | | TSSS | TSP[3:0] | | | | TMTSCE | EEFE | — | DCS | MME | DRE | DCE | TPRI |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 21.22 RSCANnGCFG Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | ITRCP[15:0] | Interval Timer Prescaler Set<br>When these bits are set to M, the pclk is divided by M.<br>Setting $0000_H$ is prohibited when the interval timer is in use. |
| 15 to 13 | TSBTCS[2:0] | Timestamp Clock Source Select<br>b15 b14 b13<br>0 0 0: Channel 0 bit time clock<br>0 0 1: Channel 1 bit time clock<br>0 1 0: Channel 2 bit time clock<br>0 1 1: Channel 3 bit time clock<br>1 0 0: Channel 4 bit time clock<br>1 0 1: Channel 5 bit time clock<br>1 1 0: Setting prohibited<br>1 1 1: Setting prohibited |
| 12 | TSSS | Timestamp Source Select<br>0: pclk/2*[1]<br>1: Bit time clock |

**Table 21.22 RSCANnGCFG Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 to 8 | TSP[3:0] | Timestamp Clock Source Division<br>b11 b10 b9 b8<br>0 0 0 0: Not divided<br>0 0 0 1: Divided by 2<br>0 0 1 0: Divided by 4<br>0 0 1 1: Divided by 8<br>0 1 0 0: Divided by 16<br>0 1 0 1: Divided by 32<br>0 1 1 0: Divided by 64<br>0 1 1 1: Divided by 128<br>1 0 0 0: Divided by 256<br>1 0 0 1: Divided by 512<br>1 0 1 0: Divided by 1024<br>1 0 1 1: Divided by 2048<br>1 1 0 0: Divided by 4096<br>1 1 0 1: Divided by 8192<br>1 1 1 0: Divided by 16384<br>1 1 1 1: Divided by 32768 |
| 7 | TMTSCE | Transmission Timestamp Enable<br>0: Transmission timestamp is disabled.<br>1: Transmission timestamp is enabled. |
| 6 | EEFE | ECC Error Flag Enable<br>0: The ECC error flag is disabled.<br>1: The ECC error flag is enabled. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 | DCS | CAN Clock Source Select*2<br>0: clkc<br>1: clk_xincan |
| 3 | MME | Mirror Function Enable<br>0: Mirror function is disabled.<br>1: Mirror function is enabled. |
| 2 | DRE | DLC Replacement Enable<br>0: DLC replacement is disabled.<br>1: DLC replacement is enabled. |
| 1 | DCE | DLC Check Enable<br>0: DLC check is disabled.<br>1: DLC check is enabled. |
| 0 | TPRI | Transmit Priority Select<br>0: ID priority<br>1: Transmit buffer number priority |

Note 1. When specifying pclk/2 as the timestamp counter count source, set bits TSBTCS[2:0] to $000_B$.

Note 2. For the CAN clock frequency settings, see **Table 21.8, Range of Operating Frequency Depending on the Transfer Rate and the Number of Channels in Use in the RH850/F1K**.

Modify the RSCANnGCFG register only in global reset mode.

### ITRCP[15:0] Bits

These bits are used to set a clock source division value of the interval timer for FIFO buffers. See **Section 21.9.3.1, Interval Transmission Function**.

### TSBTCS[2:0] Bits

When the TSSS bit is 1, these bits are used to select the channel of the bit time clock that will be the clock source of the timestamp counter.

#### TSSS Bit

This bit is used to select a clock source of the timestamp counter.

#### TSP[3:0] Bits

A clock obtained by dividing the clock source selected with the TSBTCS[2:0] bits and TSSS bit according to the TSP[3:0] bits is used as the timestamp counter count source.

#### TMTSCE Bit

Setting this bit to 1 makes it possible to store the timestamp of a transmitted message in the transmit history buffer. The timestamp is stored in TMTS[15:0] bits in the RSCANnTHLACCm register.

#### EEFE Bit

Setting this bit to 1 sets the EEFm bit in the RSCANnGERFL register to 1 when a 2-bit ECC error is detected during the transmission priority determination. At this time, the message in which a 2-bit ECC error was detected is not transmitted.

#### DCS Bit

When this bit is set to 0, clkc is used as the clock source of the CAN clock (fCAN).

When this bit is set to 1, clk_xincan is used as the clock source of the CAN clock (fCAN).

For the CAN clock frequency settings, see **Table 21.8, Range of Operating Frequency Depending on the Transfer Rate and the Number of Channels in Use in the RH850/F1K**.

#### MME Bit

Setting this bit to 1 makes the mirror function available.

#### DRE Bit

When the DRE bit is set to 1, the DLC value of the receive rule is stored in the buffer instead of the DLC value of the received message after the DLC value has passed through the DLC filter. In this case, a value of $00_H$ is stored in each data byte beyond the DLC value of the receive rule.

The DLC replacement function is only available when the DCE bit is set to 1 (DLC check is enabled).

#### DCE Bit

Setting this bit to 1 makes the DLC check function available. When disabling the DLC check function, set the GAFLDLC[3:0] bits in the RSCANnGAFLP0j register to $0000_B$ before clearing the DCE bit in the RSCANnGCFG register to 0.

#### TPRI Bit

This bit is used to set the transmit priority.

When this bit is set to 0, ID priority is selected and the transmit priority complies with the CAN bus arbitration rule (ISO11898-1 specifications). When this bit is set to 1, transmit buffer number priority is selected and the lowest transmit buffer number of those has the highest priority.

While the transmit queue is in use, this bit should be set to 0.

### 21.4.3.2 RSCANnGCTR — Global Control Register

**Access:** RSCANnGCTR register can be read or written in 32-bit units
RSCANnGCTRL, RSCANnGCTRH registers can be read or written in 16-bit units
RSCANnGCTRLL, RSCANnGCTRLH, RSCANnGCTRHL registers can be read or written in 8-bit units

**Address:** RSCANnGCTR: <RSCANn_base> + $0088_H$

RSCANnGCTRL: <RSCANn_base> + $0088_H$,
RSCANnGCTRH: <RSCANn_base> + $008A_H$

RSCANnGCTRLL: <RSCANn_base> + $0088_H$,
RSCANnGCTRLH: <RSCANn_base> + $0089_H$,
RSCANnGCTRHL: <RSCANn_base> + $008A_H$

**Value after reset:** $0000\ 0005_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TSRST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | THLEIE | MEIE | DEIE | — | — | — | — | — | GSLPR | GMDC[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

**Table 21.23 RSCANnGCTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 17 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 16 | TSRST | Timestamp Counter Reset<br>Setting the TSRST bit to 1 resets the timestamp counter. This bit is always read as 0. |
| 15 to 11 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 10 | THLEIE | Transmit History Buffer Overflow Interrupt Enable<br>0: Transmit history buffer overflow interrupt is disabled.<br>1: Transmit history buffer overflow interrupt is enabled. |
| 9 | MEIE | FIFO Message Lost Interrupt Enable<br>0: FIFO message lost interrupt is disabled.<br>1: FIFO message lost interrupt is enabled. |
| 8 | DEIE | DLC Error Interrupt Enable<br>0: DLC error interrupt is disabled.<br>1: DLC error interrupt is enabled. |
| 7 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 | GSLPR | Global Stop Mode<br>0: Other than global stop mode<br>1: Global stop mode |
| 1, 0 | GMDC[1:0] | Global Mode Select<br>b1 b0<br>0 0: Global operating mode<br>0 1: Global reset mode<br>1 0: Global test mode<br>1 1: Setting prohibited |

#### TSRST Bit

This bit is used to reset the timestamp counter. When this bit is set to 1, the RSCANnGTSC register is cleared to $0000_H$.

#### THLEIE Bit

When the THLEIE bit is set to 1 and the THLES flag in the RSCANnGERFL register is set to 1, an interrupt request is generated. Modify this bit only in global reset mode.

#### MEIE Bit

When the MEIE bit is set to 1 and the MES flag in the RSCANnGERFL register is set to 1, an interrupt request is generated. Modify this bit only in global reset mode.

#### DEIE Bit

When the DEIE bit is set to 1 and the DEF flag in the RSCANnGERFL register is set to 1, an interrupt request is generated. Modify this bit only in global reset mode.

#### GSLPR Bit

Setting this bit to 1 places the RS-CANFD module into global stop mode.
Clearing this bit to 0 makes the RS-CANFD module leave from global stop mode.
This bit should not be modified in global operating mode or global test mode.

#### GMDC[1:0] Bits

These bits are used to select the mode of entire RS-CANFD module (global operating mode, global reset mode, or global test mode). For details, see **Section 21.7.1, Global Modes**. Setting the GSLPR bit to 1 when in global reset mode places the RS-CANFD module into global stop mode.

### 21.4.3.3 RSCANnGSTS — Global Status Register

**Access:** RSCANnGSTS register is a read-only register that can be read in 32-bit units
RSCANnGSTSL register is a read-only register that can be read in 16-bit units
RSCANnGSTSLL register is a read-only register that can be read in 8-bit units

**Address:** RSCANnGSTS: <RSCANn_base> + $008C_H$

RSCANnGSTSL: <RSCANn_base> + $008C_H$

RSCANnGSTSLL: <RSCANn_base> + $008C_H$

**Value after reset:** $0000\ 000D_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | GRAMINIT | GSLPSTS | GHLTSTS | GRSTSTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.24 RSCANnGSTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is returned. |
| 3 | GRAMINIT | CAN RAM Initialization Status Flag<br>0: CAN RAM initialization is completed.<br>1: CAN RAM initialization is ongoing. |
| 2 | GSLPSTS | Global Stop Status Flag<br>0: Not in global stop mode<br>1: In global stop mode |
| 1 | GHLTSTS | Global Test Status Flag<br>0: Not in global test mode<br>1: In global test mode |
| 0 | GRSTSTS | Global Reset Status Flag<br>0: Not in global reset mode<br>1: In global reset mode |

#### GRAMINIT Flag

This flag indicates the initialization status of the CAN RAM.

This flag is set to 1 after the MCU has been reset, and is cleared to 0 when CAN RAM initialization is completed.

#### GSLPSTS Flag

This flag is set to 1 when the RS-CANFD module has transitioned to global stop mode, and is cleared to 0 when the RS-CANFD module has returned from global stop mode.

#### GHLTSTS Flag

This flag is set to 1 when the RS-CANFD module has transitioned to global test mode, and is cleared to 0 when the RS-CANFD module has exited global test mode.

**GRSTSTS Flag**

This flag is set to 1 when the RS-CANFD module has transitioned to global reset mode, and is cleared to 0 when the RS-CANFD module has exited global reset mode. This flag remains 1 even when the RS-CANFD module has transitioned from global reset mode to global stop mode.

### 21.4.3.4 RSCANnGERFL — Global Error Flag Register

**Access:** RSCANnGERFL register can be read or written in 32-bit units
RSCANnGERFLL, RSCANnGERFLH registers can be read or written in 16-bit units
RSCANnGERFLLL, RSCANnGERFLHL registers can be read or written in 8-bit units

**Address:** RSCANnGERFL: <RSCANn_base> + $0090_H$

RSCANnGERFLL: <RSCANn_base> + $0090_H$,
RSCANnGERFLH: <RSCANn_base> + $0092_H$

RSCANnGERFLLL: <RSCANn_base> + $0090_H$,
RSCANnGERFLHL: <RSCANn_base> + $0092_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | EEF5 | EEF4 | EEF3 | EEF2 | EEF1 | EEF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | THLES | MES | DEF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W*1 |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.25 RSCANnGERFL Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 22, 15, 14, 7, 6, 4, 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 21 | EEF5 | ECC Error Flag for Channel 5<br>0: No 2-bit ECC error when deciding transmission priority<br>1: A 2-bit ECC error when deciding transmission priority |
| 20 | EEF4 | ECC Error Flag for Channel 4<br>0: No 2-bit ECC error when deciding transmission priority<br>1: A 2-bit ECC error when deciding transmission priority |
| 19 | EEF3 | ECC Error Flag for Channel 3<br>0: No 2-bit ECC error when deciding transmission priority<br>1: A 2-bit ECC error when deciding transmission priority |
| 18 | EEF2 | ECC Error Flag for Channel 2<br>0: No 2-bit ECC error when deciding transmission priority<br>1: A 2-bit ECC error when deciding transmission priority |
| 17 | EEF1 | ECC Error Flag for Channel 1<br>0: No 2-bit ECC error when deciding transmission priority<br>1: A 2-bit ECC error when deciding transmission priority |
| 16 | EEF0 | ECC Error Flag for Channel 0<br>0: No 2-bit ECC error when deciding transmission priority<br>1: A 2-bit ECC error when deciding transmission priority |
| 13 to 8, 5 | Reserved | When read, the undefined value is returned. When writing, write the value after reset. |
| 2 | THLES | Transmit History Buffer Overflow Status Flag<br>0: No transmit history buffer overflow has occurred.<br>1: A transmit history buffer overflow has occurred. |
| 1 | MES | FIFO Message Lost Status Flag<br>0: No FIFO message lost error has occurred.<br>1: A FIFO message lost error has occurred. |

Table 21.25 RSCANnGERFL Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | DEF | DLC Error Flag<br>0: No DLC error has occurred.<br>1: A DLC error has occurred. |

All flags in the RSCANnGERFL register are cleared to 0 in global reset mode.

### EEFm Flag

While the EEFE bit in the RSCANnGCFG register is 1, when a 2-bit ECC error is detected during the transmission priority determination of channel m (m = 0 to 5), the EEFm flag is set to 1, disabling message transmission. This flag can be cleared to 0 by writing 0 by the program.

### THLES Flag

The THLES flag is set to 1 when any one of the THLELT flags in the RSCANnTHLSTSm register (m = 0 to 5) is set to 1.

This flag is cleared to 0 when the THLELT flags of all channels are set to 0.

### MES Flag

The MES flag is set to 1 when any one of the RFMLT flags in the RSCANnRFSTSx register (x = 0 to 7) or the CFMLT flags in the RSCANnCFSTSk register (k = 0 to 17) is set to 1.

This flag is cleared to 0 when all RFMLT flags and CFMLT flags are set to 0.

### DEF Flag

The DEF flag is set to 1 when an error has been detected during the DLC check. The program can clear this flag by writing 0 to this bit.

**NOTE**

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared. When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

### 21.4.3.5 RSCANnGTSC — Global Timestamp Counter Register

**Access:** RSCANnGTSC register is a read-only register that can be read in 32-bit units.
RSCANnGTSCL register is a read-only register that can be read in 16-bit units.

**Address:** RSCANnGTSC: <RSCANn_base> + $0094_H$

RSCANnGTSCL: <RSCANn_base> + $0094_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.26 RSCANnGTSC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. |
| 15 to 0 | TS[15:0] | Timestamp Value<br>The timestamp counter value can be read.<br>Counter Value: $0000_H$ to $FFFF_H$ |

#### TS[15:0] Bits

When the TS[15:0] bits are read, the read value shows the timestamp counter (16-bit free-running counter) value at that time. When the SOF is detected, the TS[15:0] value is captured and later stored in the receive buffer or the FIFO buffer.

Furthermore, while the TMTSCE bit in the RSCANnGCFG register is 1, the TS[15:0] value is stored in the transmit history buffer. The timestamp counter is initialized in global reset mode.

The timestamp counter start timing and stop timing depend on the count source.

- When the TSSS bit in the RSCANnGCFG register is 0 (pclk):
  The timestamp counter starts counting when the RS-CANFD module has transitioned to global operating mode.
  This counter stops counting when the RS-CANFD module has transitioned to global stop mode or global test mode.
- When the TSSS bit is 1 (CANm bit time clock):
  The timestamp counter starts counting when the corresponding channel has transitioned to channel communication mode.
  This counter stops counting when the corresponding channel has transitioned to channel reset mode or channel halt mode.

### 21.4.3.6 RSCANnGTINTSTS0 — Global TX Interrupt Status Register 0

**Access:** RSCANnGTINTSTS0 register is a read-only register that can be read in 32-bit units
RSCANnGTINTSTS0L, RSCANnGTINTSTS0H registers are read-only registers that can be read in 16-bit units
RSCANnGTINTSTS0LL, RSCANnGTINTSTS0LH, RSCANnGTINTSTS0HL, RSCANnGTINTSTS0HH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnGTINTSTS0: <RSCANn_base> + $0460_H$

RSCANnGTINTSTS0L: <RSCANn_base> + $0460_H$,
RSCANnGTINTSTS0H: <RSCANn_base> + $0462_H$

RSCANnGTINTSTS0LL: <RSCANn_base> + $0460_H$,
RSCANnGTINTSTS0LH: <RSCANn_base> + $0461_H$,
RSCANnGTINTSTS0HL: <RSCANn_base> + $0462_H$,
RSCANnGTINTSTS0HH: <RSCANn_base> + $0463_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | THIF3 | CFTIF3 | TQIF3 | TAIF3 | TSIF3 | — | — | — | THIF2 | CFTIF2 | TQIF2 | TAIF2 | TSIF2 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R*1 | R*1 | R*1 | R*1 | R*1 | R | R | R | R*1 | R*1 | R*1 | R*1 | R*1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | THIF1 | CFTIF1 | TQIF1 | TAIF1 | TSIF1 | — | — | — | THIF0 | CFTIF0 | TQIF0 | TAIF0 | TSIF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R*1 | R*1 | R*1 | R*1 | R*1 | R | R | R | R*1 | R*1 | R*1 | R*1 | R*1 |

Note 1. This bit is automatically cleared in the global reset or channel reset mode.

**Table 21.27 RSCANnGTINTSTS0 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | When read, the value after reset is returned. |
| 28 | THIF3 | Channel 3 Transmit History Interrupt Status Flag<br>0: Transmit history interrupt is not requested.<br>1: Transmit history interrupt is requested. |
| 27 | CFTIF3 | Channel 3 Transmit/receive FIFO Transmit Interrupt Status Flag<br>0: Transmit/receive FIFO transmit interrupt is not requested.<br>1: Transmit/receive FIFO transmit interrupt is requested. |
| 26 | TQIF3 | Channel 3 Transmit Queue Interrupt Status Flag<br>0: Transmit queue interrupt is not requested.<br>1: Transmit queue interrupt is requested. |
| 25 | TAIF3 | Channel 3 Transmit Buffer Abort Interrupt Status Flag<br>0: Transmit buffer abort interrupt is not requested.<br>1: Transmit buffer abort interrupt is requested |
| 24 | TSIF3 | Channel 3 Transmit Buffer Transmit Complete Interrupt Status Flag<br>0: Transmit buffer transmit complete interrupt is not requested.<br>1: Transmit buffer transmit complete interrupt is requested. |
| 23 to 21 | Reserved | When read, the value after reset is returned. |
| 20 | THIF2 | Channel 2 Transmit History Interrupt Status Flag<br>0: Transmit history interrupt is not requested.<br>1: Transmit history interrupt is requested. |
| 19 | CFTIF2 | Channel 2 Transmit/receive FIFO Transmit Interrupt Status Flag<br>0: Transmit/receive FIFO transmit interrupt is not requested.<br>1: Transmit/receive FIFO transmit interrupt is requested. |
| 18 | TQIF2 | Channel 2 Transmit Queue Interrupt Status Flag<br>0: Transmit queue interrupt is not requested.<br>1: Transmit queue interrupt is requested. |

Table 21.27 RSCANnGTINTSTS0 Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 17 | TAIF2 | Channel 2 Transmit Buffer Abort Interrupt Status Flag<br>0: Transmit buffer abort interrupt is not requested.<br>1: Transmit buffer abort interrupt is requested. |
| 16 | TSIF2 | Channel 2 Transmit Buffer Interrupt Status Flag<br>0: Transmit buffer transmit complete interrupt is not requested.<br>1: Transmit buffer transmit complete interrupt is requested. |
| 15 to 13 | Reserved | When read, the value after reset is returned. |
| 12 | THIF1 | Channel 1 Transmit History Interrupt Status Flag<br>0: Transmit history interrupt is not requested.<br>1: Transmit history interrupt is requested. |
| 11 | CFTIF1 | Channel 1 Transmit/receive FIFO Transmit Interrupt Status Flag<br>0: Transmit/receive FIFO transmit interrupt is not requested.<br>1: Transmit/receive FIFO transmit interrupt is requested. |
| 10 | TQIF1 | Channel 1 Transmit Queue Interrupt Status Flag<br>0: Transmit queue interrupt is not requested.<br>1: Transmit queue interrupt is requested. |
| 9 | TAIF1 | Channel 1 Transmit Buffer Abort Interrupt Status Flag<br>0: Transmit buffer abort interrupt is not requested.<br>1: Transmit buffer abort interrupt is requested. |
| 8 | TSIF1 | Channel 1 Transmit Buffer Interrupt Status Flag<br>0: Transmit buffer transmit complete interrupt is not requested.<br>1: Transmit buffer transmit complete interrupt is requested. |
| 7 to 5 | Reserved | When read, the value after reset is returned. |
| 4 | THIF0 | Channel 0 Transmit History Interrupt Status Flag<br>0: Transmit history interrupt is not requested.<br>1: Transmit history interrupt is requested. |
| 3 | CFTIF0 | Channel 0 Transmit/receive FIFO Transmit Interrupt Status Flag<br>0: Transmit/receive FIFO transmit interrupt is not requested.<br>1: Transmit/receive FIFO transmit interrupt is requested. |
| 2 | TQIF0 | Channel 0 Transmit Queue Interrupt Status Flag<br>0: Transmit queue interrupt is not requested.<br>1: Transmit queue interrupt is requested. |
| 1 | TAIF0 | Channel 0 Transmit Buffer Abort Interrupt Status Flag<br>0: Transmit buffer abort interrupt is not requested.<br>1: Transmit buffer abort interrupt is requested. |
| 0 | TSIF0 | Channel 0 Transmit Buffer Interrupt Status Flag<br>0: Transmit buffer transmit complete interrupt is not requested.<br>1: Transmit buffer transmit complete interrupt is requested. |

### TSIFm Bits

The TSIFm bit is set to 1 when the TMIEp bit in the RSCANnTMIECy register is set to 1 (transmit buffer interrupt enabled) and the corresponding TMTRF[1:0] flags in the RSCAN0TMSTSp register are set to $10_B$ (transmit completed without abort request) or $11_B$ (transmit completed with abort request).

When all TMTRF[1:0] flags that meet the condition to set the TSIFm bit to 1 are cleared to $00_B$, this flag is cleared to 0. In addition, clearing the TMIEp bit to 0 also clears this flag to 0.

### TAIFm Bits

The TAIFm bit is set to 1 when the TAIE bit in the RSCANnCmCTR register is 1 (transmit abort interrupt enabled) and the TMTRF[1:0] flags in the RSCAN0TMSTSp register are set to $01_B$ (transmit abort completed).

This flag is cleared to 0 when all TMTRF[1:0] flags that completed transmit abort are cleared to $00_B$.

### TQIFm Bits

When the TXQIE bit in the RSCANnTXQCCm register is set to 1 (transmit queue interrupt enabled) and the TXQIF bit in the RSCANnTXQSTSm register is set to 1 (transmit queue interrupt request), the TQIFm bit is set to 1.

When the TXQIF bit (transmit queue interrupt request) in the RSCANnTXQSTSm register is cleared to 0, this bit is cleared to 0. This flag is also cleared to 0 when the TXQIE bit is cleared to 0.

### CFTIFm Bits

When the CFTXIE bit in the RSCANnCFCCk register is set to 1 (transmit/receive FIFO transmit interrupt enabled) and the CFTXIF bit in the RSCANnCFSTSk register is set to 1 (transmit/receive FIFO transmit interrupt request), the CFTIFm bit is set to 1.

When all CFTXIF bits that meet the condition to set the CFTIFm bit to 1 are cleared to 0, this bit is cleared to 0. This flag is also cleared to 0 when the CFTXIE bit is cleared to 0.

### THIFm Bits

When the THLIE bit in the RSCANnTHLCCm register is set to 1 (transmit history interrupt enabled) and the THLIF bit in the RSCANnTHLSTSm register is set to 1 (transmit history interrupt request), the THIFm bit is set to 1.

When the THLIF bit in the RSCANnTHLSTSm register is cleared to 0, this bit is cleared to 0. This flag is also cleared to 0 when the THLIE bit is cleared to 0.

### 21.4.3.7 RSCANnGTINTSTS1 — Global TX Interrupt Status Register 1

**Access:** RSCANnGTINTSTS1 register is a read-only register that can be read in 32-bit units
RSCANnGTINTSTS1L register is a read-only register that can be read in 16-bit units
RSCANnGTINTSTS1LL, RSCANnGTINTSTS1LH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnGTINTSTS1: <RSCANn_base> + $0464_H$

RSCANnGTINTSTS1L: <RSCANn_base> + $0464_H$

RSCANnGTINTSTS1LL: <RSCANn_base> + $0464_H$,
RSCANnGTINTSTS1LH: <RSCANn_base> + $0465_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | THIF5 | CFTIF5 | TQIF5 | TAIF5 | TSIF5 | — | — | — | THIF4 | CFTIF4 | TQIF4 | TAIF4 | TSIF4 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R*1 | R*1 | R*1 | R*1 | R*1 | R | R | R | R*1 | R*1 | R*1 | R*1 | R*1 |

Note 1. This bit is automatically cleared in the global reset or channel reset mode.

**Table 21.28 RSCANnGTINTSTS1 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 13 | Reserved | When read, the value after reset is returned. |
| 12 | THIF5 | Channel 5 Transmit History Interrupt Status Flag<br>0: Transmit history interrupt is not requested.<br>1: Transmit history interrupt is requested. |
| 11 | CFTIF5 | Channel 5 Transmit/receive FIFO Transmit Interrupt Status Flag<br>0: Transmit/receive FIFO transmit interrupt is not requested.<br>1: Transmit/receive FIFO transmit interrupt is requested. |
| 10 | TQIF5 | Channel 5 Transmit Queue Interrupt Status Flag<br>0: Transmit queue interrupt is not requested.<br>1: Transmit queue interrupt is requested. |
| 9 | TAIF5 | Channel 5 Transmit Buffer Abort Interrupt Status Flag<br>0: Transmit buffer abort interrupt is not requested.<br>1: Transmit buffer abort interrupt is requested. |
| 8 | TSIF5 | Channel 5 Transmit Buffer Interrupt Status Flag<br>0: Transmit buffer transmit complete interrupt is not requested.<br>1: Transmit buffer transmit complete interrupt is requested. |
| 7 to 5 | Reserved | When read, the value after reset is returned. |
| 4 | THIF4 | Channel 4 Transmit History Interrupt Status Flag<br>0: Transmit history interrupt is not requested.<br>1: Transmit history interrupt is requested. |
| 3 | CFTIF4 | Channel 4 Transmit/receive FIFO Transmit Interrupt Status Flag<br>0: Transmit/receive FIFO transmit interrupt is not requested.<br>1: Transmit/receive FIFO transmit interrupt is requested. |
| 2 | TQIF4 | Channel 4 Transmit Queue Interrupt Status Flag<br>0: Transmit queue interrupt is not requested.<br>1: Transmit queue interrupt is requested. |
| 1 | TAIF4 | Channel 4 Transmit Buffer Abort Interrupt Status Flag<br>0: Transmit buffer abort interrupt is not requested.<br>1: Transmit buffer abort interrupt is requested. |

Table 21.28 RSCANnGTINTSTS1 Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | TSIF4 | Channel 4 Transmit Buffer Interrupt Status Flag<br>0: Transmit buffer transmit complete interrupt is not requested.<br>1: Transmit buffer transmit complete interrupt is requested. |

### TSIFm Bits

The TSIFm bit is set to 1 when the TMIEp bit in the RSCANnTMIECy register is set to 1 (transmit buffer interrupt enabled) and the corresponding TMTRF[1:0] flags in the RSCAN0TMSTSp register are set to $10_B$ (transmit completed without abort request) or $11_B$ (transmit completed with abort request).

When all TMTRF[1:0] flags that meet the condition to set the TSIFm bit to 1 are cleared to $00_B$, this flag is cleared to 0. In addition, clearing the TMIEp bit to 0 also clears this flag to 0.

### TAIFm Bits

The TAIFm bit is set to 1 when the TAIE bit in the RSCANnCmCTR register is set to 1 (transmit abort interrupt enabled) and the TMTRF[1:0] flags in the RSCANnTMSTSp register are set to $01_B$ (transmit abort completed).

This flag is cleared to 0 when all TMTRF[1:0] flags that completed transmit abort are cleared to $00_B$.

### TQIFm Bits

When the TXQIE bit in the RSCANnTXQCCm register is set to 1 (transmit queue interrupt enabled) and the TXQIF bit in the RSCANnTXQSTSm register is set to 1 (transmit queue interrupt request), the TQIFm bit is set to 1.

When the TXQIF bit (transmit queue interrupt request) in the RSCANnTXQSTSm register is cleared to 0, this bit is cleared to 0. Clearing the TXQIE bit to 0 also clears this flag to 0.

### CFTIFm Bits

When the CFTXIE bit in the RSCANnCFCCk register is set to 1 (transmit/receive FIFO transmit interrupt enabled) and the CFTXIF bit in the RSCANnCFSTSk register is set to 1 (transmit/receive FIFO transmit interrupt request), the CFTIFm bit is set to 1.

When all CFTXIF bits that meet the condition to set the CFTIFm bit to 1 are cleared to 0, this bit is cleared to 0. This flag is also cleared to 0 when the CFTXIE bit is cleared to 0.

### THIFm Bits

When the THLIE bit in the RSCANnTHLCCm register is set to 1 (transmit history interrupt enabled) and the THLIF bit in the RSCANnTHLSTSm register is set to 1 (transmit history interrupt request), the THIFm bit is set to 1.

When the THLIF bit in the RSCANnTHLSTSm register is cleared to 0, this bit is cleared to 0.

This flag is also cleared to 0 when the THLIE bit is cleared to 0.

### 21.4.3.8 RSCANnGFDCFG — Global FD Configuration Register

**Access:** RSCANnGFDCFG register can be read or written in 32-bit unit.
RSCANnGFDCFGL register can be read or written in 16-bit unit.
RSCANnGFDCFGLH register can be read or written in 8-bit unit.

**Address:** RSCANnGFDCFG: <RSCANn_base> + $0474_H$

RSCANnGFDCFGL: <RSCANn_base> + $0474_H$

RSCANnGFDCFGLH: <RSCANn_base> + $0475_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TSCCFG[1:0] | | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R | R |

**Table 21.29 RSCANnGFDCFG Register Content**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 10 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 9, 8 | TSCCFG[1:0] | Timestamp Capture Setting<br>b9 b8<br>0 0: Captured at a sample point in the SOF bit.<br>0 1: Captured when a valid frame has been transmitted/received.<br>1 0: Setting prohibited<br>1 1: Setting prohibited |
| 7 to 1 | Reserved | These bits are read as the value after reset.<br>The write value should be the value after reset. |
| 0 | Reserved | When read, an undefined value is returned.<br>The write value should be the value after reset. |

#### TSCCFG[1:0] Bit

Select a point where a timestamp value is captured. Modify this bit only in global reset mode.

## 21.4.4 Details of Receive Rule-related Registers

### 21.4.4.1 RSCANnGAFLECTR — Receive Rule Entry Control Register

**Access:** RSCANnGAFLECTR register can be read or written in 32-bit units
RSCANnGAFLECTRL register can be read or written in 16-bit units
RSCANnGAFLECTRLL, RSCANnGAFLECTRLH registers can be read or written in 8-bit units

**Address:** RSCANnGAFLECTR: <RSCANn_base> + $0098_H$

RSCANnGAFLECTRL: <RSCANn_base> + $0098_H$

RSCANnGAFLECTRLL: <RSCANn_base> + $0098_H$,
RSCANnGAFLECTRLH: <RSCANn_base> + $0099_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | AFLDAE | — | — | — | AFLPN[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 21.30 RSCANnGAFLECTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 8 | AFLDAE | Receive Rule Table Write Enable<br>0: Receive rule table write is disabled.<br>1: Receive rule table write is enabled. |
| 7 to 5 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 4 to 0 | AFLPN[4:0] | Receive Rule Table Page Number Configuration<br>A page number can be selected in the range of page 0 ($00000_B$) to page 23 ($10111_B$). |

#### AFLDAE Bit

Setting this bit to 0 disables the write to the receive rule table. After writes to the receive rule table are completed, set this bit to 0 to disable the write to the table. The receive rule table can be read regardless of the value of this bit.

Set the AFLDAE bit to 1 only in global reset mode.

#### AFLPN[4:0] Bits

These bits are used to set the page number of the receive rule table. Sixteen receive rules can be set per page.

Set these bits to a value within the range of $00000_B$ to $10111_B$.

### 21.4.4.2 RSCANnGAFLCFG0 — Receive Rule Configuration Register 0

**Access:** RSCANnGAFLCFG0 register can be read or written in 32-bit units
RSCANnGAFLCFG0L, RSCANnGAFLCFG0H registers can be read or written in 16-bit units
RSCANnGAFLCFG0LL, RSCANnGAFLCFG0LH, RSCANnGAFLCFG0HL, RSCANnGAFLCFG0HH registers can be read or written in 8-bit units

**Address:** RSCANnGAFLCFG0: <RSCANn_base> + $009C_H$

RSCANnGAFLCFG0L: <RSCANn_base> + $009C_H$,
RSCANnGAFLCFG0H: <RSCANn_base> + $009E_H$

RSCANnGAFLCFG0LL: <RSCANn_base> + $009C_H$,
RSCANnGAFLCFG0LH: <RSCANn_base> + $009D_H$,
RSCANnGAFLCFG0HL: <RSCANn_base> + $009E_H$,
RSCANnGAFLCFG0HH: <RSCANn_base> + $009F_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RNC0[7:0] | | | | | | | | RNC1[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RNC2[7:0] | | | | | | | | RNC3[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.31 RSCANnGAFLCFG0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RNC0[7:0] | Number of Rules for Channel 0<br>Set the number of receive rules exclusively used for channel 0. |
| 23 to 16 | RNC1[7:0] | Number of Rules for Channel 1<br>Set the number of receive rules exclusively used for channel 1. |
| 15 to 8 | RNC2[7:0] | Number of Rules for Channel 2<br>Set the number of receive rules exclusively used for channel 2. |
| 7 to 0 | RNC3[7:0] | Number of Rules for Channel 3<br>Set the number of receive rules exclusively used for channel 3. |

Modify the RSCANnGAFLCFG0 register only in global reset mode.

Up to 64 × (number of channels) rules can be registered in the receive rule table as the entire unit. The number of receive rules per channel should meet the following conditions.

- The maximum number of rules per channel is 128.
- The total of the number of rules allocated to each channel is not larger than the number of rules that can be registered in the entire unit.

#### RNC0[7:0] Bits

These bits are used to set the number of rules to be registered in the channel 0 receive rule table.

Set these bits to a value within the range of $00_H$ to $80_H$.

#### RNC1[7:0] Bits

These bits are used to set the number of rules to be registered in the channel 1 receive rule table.

Set these bits to a value within the range of $00_H$ to $80_H$.

**RNC2[7:0] Bits**

These bits are used to set the number of rules to be registered in the channel 2 receive rule table.

Set these bits to a value within the range of $00_H$ to $80_H$.

**RNC3[7:0] Bits**

These bits are used to set the number of rules to be registered in the channel 3 receive rule table.

Set these bits to a value within the range of $00_H$ to $80_H$.

### 21.4.4.3 RSCANnGAFLCFG1 — Receive Rule Configuration Register 1

**Access:** RSCANnGAFLCFG1 register can be read or written in 32-bit units
RSCANnGAFLCFG1H register can be read or written in 16-bit units
RSCANnGAFLCFG1HL, RSCANnGAFLCFG1HH registers can be read or written in 8-bit units

**Address:** RSCANnGAFLCFG1: <RSCANn_base> + $00A0_H$

RSCANnGAFLCFG1H: <RSCANn_base> + $00A2_H$

RSCANnGAFLCFG1HL: <RSCANn_base> + $00A2_H$,
RSCANnGAFLCFG1HH: <RSCANn_base> + $00A3_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RNC4[7:0] | | | | | | | | RNC5[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.32 RSCANnGAFLCFG1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RNC4[7:0] | Number of Rules for Channel 4<br>Set the number of receive rules exclusively used for channel 4. |
| 23 to 16 | RNC5[7:0] | Number of Rules for Channel 5<br>Set the number of receive rules exclusively used for channel 5. |
| 15 to 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Modify the RSCANnGAFLCFG1 register only in global reset mode.

Up to 64 × (number of channels) rules can be registered in the receive rule table as the entire unit. The number of receive rules per channel should meet the following conditions.

- The maximum number of rules per channel is 128.
- The total of the number of rules allocated to each channel is not larger than the number of rules that can be registered in the entire unit.

#### RNC4[7:0] Bits

These bits are used to set the number of rules to be registered in the channel 4 receive rule table.

Set these bits to a value within the range of $00_H$ to $80_H$.

#### RNC5[7:0] Bits

These bits are used to set the number of rules to be registered in the channel 5 receive rule table.

Set these bits to a value within the range of $00_H$ to $80_H$.

### 21.4.4.4 RSCANnGAFLIDj — Receive Rule ID Register (j = 0 to 15)

**Access:** RSCANnGAFLIDj register can be read or written in 32-bit units
RSCANnGAFLIDjL, RSCANnGAFLIDjH registers can be read or written in 16-bit units
RSCANnGAFLIDjLL, RSCANnGAFLIDjLH, RSCANnGAFLIDjHL, RSCANnGAFLIDjHH registers can be read or written in 8-bit units

**Address:** RSCANnGAFLIDj: <RSCANn_base> + $0500_H$ + ($10_H$ × j)

RSCANnGAFLIDjL: <RSCANn_base> + $0500_H$ + ($10_H$ × j),
RSCANnGAFLIDjH: <RSCANn_base> + $0502_H$ + ($10_H$ × j)

RSCANnGAFLIDjLL: <RSCANn_base> + $0500_H$ + ($10_H$ × j),
RSCANnGAFLIDjLH: <RSCANn_base> + $0501_H$ + ($10_H$ × j),
RSCANnGAFLIDjHL: <RSCANn_base> + $0502_H$ + ($10_H$ × j),
RSCANnGAFLIDjHH: <RSCANn_base> + $0503_H$ + ($10_H$ × j)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLIDE | GAFLRTR | GAFLLB | GAFLID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.33 RSCANnGAFLIDj Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | GAFLIDE | IDE Select<br>0: Standard ID<br>1: Extended ID |
| 30 | GAFLRTR | RTR Select<br>0: Data frame<br>1: Remote frame |
| 29 | GAFLLB | Receive Rule Target Message Select<br>0: When a message transmitted from another CAN node is received<br>1: When the own transmitted message is received |
| 28 to 0 | GAFLID[28:0] | ID<br>Set the ID of the receive rule.<br>For the standard ID, set the ID in bits b10 to b0 and set bits b28 to b11 to 0. |

Modify the RSCANnGAFLIDj register when the AFLDAE bit in the RSCANnGAFLECTR register is set to 1 (receive rule table write is enabled) in global reset mode.

#### GAFLIDE Bit

This bit is used to select the ID format (standard ID or extended ID) of the receive rule. This bit is compared with the IDE bit in the received message during the acceptance filter processing.

#### GAFLRTR Bit

This bit is used to select the frame format (data frame or remote frame) of the receive rule. This bit is compared with the RTR bit in the received message during the acceptance filter processing.

#### GAFLLB Bit

When this bit is set to 0, data processing using the receive rule is performed when receiving messages transmitted from another CAN node.

When this bit is set to 1 when the mirror function is used, data processing using the receive rule is performed when the CAN node is receiving its own transmitted messages.

#### GAFLID[28:0] Bits

These bits are used to set the ID field of the receive rule. The ID value set by these bits is compared with the ID of the received message during the acceptance filter processing.

### 21.4.4.5 RSCANnGAFLMj — Receive Rule Mask Register (j = 0 to 15)

**Access:** RSCANnGAFLMj register can be read or written in 32-bit units
RSCANnGAFLMjL, RSCANnGAFLMjH registers can be read or written in 16-bit units
RSCANnGAFLMjLL, RSCANnGAFLMjLH, RSCANnGAFLMjHL, RSCANnGAFLMjHH registers can be read or written in 8-bit units

**Address:** RSCANnGAFLMj: <RSCANn_base> + $0504_H$ + ($10_H$ × j)

RSCANnGAFLMjL: <RSCANn_base> + $0504_H$ + ($10_H$ × j),
RSCANnGAFLMjH: <RSCANn_base> + $0506_H$ + ($10_H$ × j)

RSCANnGAFLMjLL: <RSCANn_base> + $0504_H$ + ($10_H$ × j),
RSCANnGAFLMjLH: <RSCANn_base> + $0505_H$ + ($10_H$ × j),
RSCANnGAFLMjHL: <RSCANn_base> + $0506_H$ + ($10_H$ × j),
RSCANnGAFLMjHH: <RSCANn_base> + $0507_H$ + ($10_H$ × j)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLIDEM | GAFLRTRM | — | GAFLIDM[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLIDM[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.34 RSCANnGAFLMj Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | GAFLIDEM | IDE Mask<br>0: The IDE bit is not compared.<br>1: The IDE bit is compared. |
| 30 | GAFLRTRM | RTR Mask<br>0: The RTR bit is not compared.<br>1: The RTR bit is compared |
| 29 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 28 to 0 | GAFLIDM[28:0] | ID Mask<br>0: The corresponding ID bit is not compared.<br>1: The corresponding ID bit is compared. |

Modify the RSCANnGAFLMj register when the AFLDAE bit in the RSCANnGAFLECTR register is set to 1 (receive rule table write is enabled) in global reset mode.

#### GAFLIDEM Bit

When this bit is set to 1, filter processing is performed only for messages of the ID format specified by the GAFLIDE bit in the RSCANnGAFLIDj register.

When this bit is cleared to 0, the IDs of all the receive messages and the specified IDs are regarded as matched. To set the GAFLIDEM bit to 0, set the GAFLIDM[28:0] bits to all 0 at the same time.

#### GAFLRTRM Bit

This bit is used to mask the RTR bit of the receive rule.

#### GAFLIDM[28:0] Bits

These bits are used to mask the corresponding ID bit of the receive rule.

### 21.4.4.6 RSCANnGAFLP0j — Receive Rule Pointer 0 Register (j = 0 to 15)

**Access:** RSCANnGAFLP0j register can be read or written in 32-bit units
RSCANnGAFLP0jL, RSCANnGAFLP0jH registers can be read or written in 16-bit units
RSCANnGAFLP0jLH, RSCANnGAFLP0jHL, RSCANnGAFLP0jHH registers can be read or written in 8-bit units

**Address:** RSCANnGAFLP0j: <RSCANn_base> + $0508_H$ + ($10_H$ × j)

RSCANnGAFLP0jL: <RSCANn_base> + $0508_H$ + ($10_H$ × j),
RSCANnGAFLP0jH: <RSCANn_base> + $050A_H$ + ($10_H$ × j)

RSCANnGAFLP0jLH: <RSCANn_base> + $0509_H$ + ($10_H$ × j),
RSCANnGAFLP0jHL: <RSCANn_base> + $050A_H$ + ($10_H$ × j),
RSCANnGAFLP0jHH: <RSCANn_base> + $050B_H$ + ($10_H$ × j)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLDLC[3:0] | | | | GAFLPTR[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLRMV | GAFLRMDP[6:0] | | | | | | | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

**Table 21.35 RSCANnGAFLP0j Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | GAFLDLC[3:0] | Receive Rule DLC<br>b31 b30 b29 b28<br>0 0 0 0: DLC check is disabled.<br>0 0 0 1: 1 data byte<br>0 0 1 0: 2 data bytes<br>0 0 1 1: 3 data bytes<br>0 1 0 0: 4 data bytes<br>0 1 0 1: 5 data bytes<br>0 1 1 0: 6 data bytes<br>0 1 1 1: 7 data bytes<br>1 X X X: 8 data bytes |
| 27 to 16 | GAFLPTR[11:0] | Receive Rule Label<br>Set the 12-bit label information. |
| 15 | GAFLRMV | Receive Buffer Enable<br>0: No receive buffer is used.<br>1: A receive buffer is used. |
| 14 to 8 | GAFLRMDP[6:0] | Receive Buffer Number Select<br>Set the receive buffer number to store receive messages. |
| 7 to 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Modify the RSCANnGAFLP0j register when the AFLDAE bit in the RSCANnGAFLECTR register is set to 1 (receive rule table write is enabled) in global reset mode.

#### GAFLDLC[3:0] Bits

These bits are used to set the minimum data length necessary for receiving messages. If the data length of a message that is being filtered is equal to or larger than the value set by the GAFLDLC[3:0] bits, the message passes the DLC check. Setting these bits to $0000_B$ disables the DLC check function allowing messages with any data length to pass the DLC check.

#### GAFLPTR[11:0] Bits

These bits are used to set a 12-bit label to be attached to messages that have passed through the filter. A label is attached when a message is stored in the receive buffer or the FIFO buffer.

#### GAFLRMV Bit

When this bit is set to 1, receive messages that have passed through the filter are stored in the receive buffer selected by the GAFLRMDP[6:0] bits.

#### GAFLRMDP[6:0] Bits

These bits are used to select the number of the receive buffer that stores receive messages that have passed through the filter when the GAFLRMV bit is set to 1. Set these bits to a value smaller than the value set by the NRXMB[7:0] bits in the RSCANnRMNB register.

### 21.4.4.7 RSCANnGAFLP1j — Receive Rule Pointer 1 Register (j = 0 to 15)

**Access:** RSCANnGAFLP1j register can be read or written in 32-bit units
RSCANnGAFLP1jL, RSCANnGAFLP1jH registers can be read or written in 16-bit units
RSCANnGAFLP1jLL, RSCANnGAFLP1jLH, RSCANnGAFLP1jHL, RSCANnGAFLP1jHH registers can be read or written in 8-bit units

**Address:** RSCANnGAFLP1j: <RSCANn_base> + $050C_H$ + ($10_H$ × j)

RSCANnGAFLP1jL: <RSCANn_base> + $050C_H$ + ($10_H$ × j),
RSCANnGAFLP1jH: <RSCANn_base> + $050E_H$ + ($10_H$ × j)

RSCANnGAFLP1jLL: <RSCANn_base> + $050C_H$ + ($10_H$ × j),
RSCANnGAFLP1jLH: <RSCANn_base> + $050D_H$ + ($10_H$ × j),
RSCANnGAFLP1jHL: <RSCANn_base> + $050E_H$ + ($10_H$ × j),
RSCANnGAFLP1jHH: <RSCANn_base> + $050F_H$ + ($10_H$ × j)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | GAFLFDP[25:16] | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLFDP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.36 RSCANnGAFLP1j Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 25 to 8 | GAFLFDP[25:8] | Transmit/Receive FIFO Buffer k Select<br>(Bit position –8 = target transmit/receive FIFO buffer number k)<br>0: Transmit/receive FIFO buffer is not selected.<br>1: Transmit/receive FIFO buffer is selected. |
| 7 to 0 | GAFLFDP[7:0] | Receive FIFO Buffer x Select<br>(Bit position = target receive FIFO buffer number x)<br>0: Receive FIFO buffer is not selected.<br>1: Receive FIFO buffer is selected. |

Modify the RSCANnGAFLP1j register when the AFLDAE bit in the RSCANnGAFLECTR register is set to 1 (receive rule table write is enabled) in global reset mode.

#### GAFLFDP[25:0] Bits

These bits are used to specify FIFO buffers that store receive messages that have passed through the filter. Up to eight FIFO buffers can be selected. However, when the GAFLRMV bit in the RSCANnGAFLP0j register is set to 1 (messages are stored in the receive buffer), up to seven FIFO buffers can be selected. Only receive FIFO buffers and the transmit/receive FIFO buffer for which the CFM[1:0] bits in the RSCANnCFCCk register are set to $00_B$ (receive mode) or $10_B$ (gateway mode) can be selected.

## 21.4.5 Details of Receive Buffer-Related Registers

### 21.4.5.1 RSCANnRMNB — Receive Buffer Number Register

**Access:** RSCANnRMNB register can be read or written in 32-bit units
RSCANnRMNBL register can be read or written in 16-bit units
RSCANnRMNBLL register can be read or written in 8-bit units

**Address:** RSCANnRMNB: <RSCANn_base> + $00A4_H$
RSCANnRMNBL: <RSCANn_base> + $00A4_H$
RSCANnRMNBLL: <RSCANn_base> + $00A4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | NRXMB[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.37 RSCANnRMNB Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 7 to 0 | NRXMB[7:0] | Receive Buffer Number Configuration<br>Set the number of receive buffers.<br>Set a value of 0 to 96. |

Modify the RSCANnRMNB register only in global reset mode.

#### NRXMB[7:0] Bits

These bits are used to set the total number of receive buffers of the RS-CANFD module. The maximum value is 16 × (number of channels).

Setting all of these bits to 0 makes receive buffers unavailable.

### 21.4.5.2 RSCANnRMNDy — Receive Buffer New Data Register (y = 0 to 2)

**Access:** RSCANnRMNDy register can be read or written in 32-bit units
RSCANnRMNDyL, RSCANnRMNDyH registers can be read or written in 16-bit units
RSCANnRMNDyLL, RSCANnRMNDyLH, RSCANnRMNDyHL, RSCANnRMNDyHH registers can be read or written in 8-bit units

**Address:** RSCANnRMNDy: <RSCANn_base> + $00A8_H$ + ($04_H$ × y)

RSCANnRMNDyL: <RSCANn_base> + $00A8_H$ + ($04_H$ × y),
RSCANnRMNDyH: <RSCANn_base> + $00AA_H$ + ($04_H$ × y)

RSCANnRMNDyLL: <RSCANn_base> + $00A8_H$ + ($04_H$ × y),
RSCANnRMNDyLH: <RSCANn_base> + $00A9_H$ + ($04_H$ × y),
RSCANnRMNDyHL: <RSCANn_base> + $00AA_H$ + ($04_H$ × y),
RSCANnRMNDyHH: <RSCANn_base> + $00AB_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMNSq (q = y × 32 + 31 to y × 32 + 16 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMNSq (q = y × 32 + 15 to y × 32 + 0 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.38 RSCANnRMNDy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | RMNSq | Receive Buffer Receive Complete Flag q (q = y × 32 + 31 to y × 32 + 16)<br>0: There is no new message in receive buffer q.<br>1: There is a new message in receive buffer q. |
| 15 to 0 | RMNSq | Receive Buffer Receive Complete Flag q (q = y × 32 + 15 to y × 32 + 0)<br>0: There is no new message in receive buffer q.<br>1: There is a new message in receive buffer q. |

Write 0 to the RSCANnRMNDy register in global operating mode or global test mode.

#### RMNSq Flags (q = 0 to 95)

Each RMNSq flag is set to 1 when the processing for storing a message in the corresponding receive buffer starts.

To clear a flag to 0, the program must write 0 to the flag. Use a store instruction to write “0” to the flag and “1” to other flags. These bits cannot be set to 0 while a message is being stored. It takes ten clock cycles of pclk to store a message.

These flags are cleared to 0 in global reset mode.

### 21.4.5.3 RSCANnRMIDq — Receive Buffer ID Register (q = 0 to 95)

**Access:** RSCANnRMIDq register is a read-only register that can be read in 32-bit units
RSCANnRMIDqL, RSCANnRMIDqH registers are read-only registers that can be read in 16-bit units
RSCANnRMIDqLL, RSCANnRMIDqLH, RSCANnRMIDqHL, RSCANnRMIDqHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnRMIDq: <RSCANn_base> + $0600_H$ + ($10_H$ × q)

RSCANnRMIDqL: <RSCANn_base> + $0600_H$ + ($10_H$ × q),
RSCANnRMIDqH: <RSCANn_base> + $0602_H$ + ($10_H$ × q)

RSCANnRMIDqLL: <RSCANn_base> + $0600_H$ + ($10_H$ × q),
RSCANnRMIDqLH: <RSCANn_base> + $0601_H$ + ($10_H$ × q),
RSCANnRMIDqHL: <RSCANn_base> + $0602_H$ + ($10_H$ × q),
RSCANnRMIDqHH: <RSCANn_base> + $0603_H$ + ($10_H$ × q)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMIDE | RMRTR | — | RMID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.39 RSCANnRMIDq Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | RMIDE | Receive Buffer IDE<br>0: Standard ID<br>1: Extended ID |
| 30 | RMRTR | Receive Buffer RTR<br>0: Data frame<br>1: Remote frame |
| 29 | Reserved | When read, the value after reset is returned. |
| 28 to 0 | RMID[28:0] | Receive Buffer ID Data<br>These bits contain the standard ID or extended ID of the received message.<br>Read bits b10 to b0 for standard ID. Bits b28 to b11 are read as 0. |

#### RMIDE Bit

This bit indicates the ID format (standard ID or extended ID) of the message stored in the receive buffer.

#### RMRTR Bit

This bit indicates the frame format (data frame or remote frame) of the message stored in the receive buffer.

#### RMID[28:0] Bits

These bits contain the ID of the message stored in the receive buffer.

### 21.4.5.4 RSCANnRMPTRq — Receive Buffer Pointer Register (q = 0 to 95)

**Access:** RSCANnRMPTRq register is a read-only register that can be read in 32-bit units
RSCANnRMPTRqL, RSCANnRMPTRqH registers are read-only registers that can be read in 16-bit units
RSCANnRMPTRqLL, RSCANnRMPTRqLH, RSCANnRMPTRqHL, RSCANnRMPTRqHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnRMPTRq: <RSCANn_base> + $0604_H$ + ($10_H$ × q)

RSCANnRMPTRqL: <RSCANn_base> + $0604_H$ + ($10_H$ × q),
RSCANnRMPTRqH: <RSCANn_base> + $0606_H$ + ($10_H$ × q)

RSCANnRMPTRqLL: <RSCANn_base> + $0604_H$ + ($10_H$ × q),
RSCANnRMPTRqLH: <RSCANn_base> + $0605_H$ + ($10_H$ × q),
RSCANnRMPTRqHL: <RSCANn_base> + $0606_H$ + ($10_H$ × q),
RSCANnRMPTRqHH: <RSCANn_base> + $0607_H$ + ($10_H$ × q)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMDLC[3:0] | | | | RMPTR[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMTS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.40 RSCANnRMPTRq Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | RMDLC[3:0] | Receive Buffer DLC Data<br>b31 b30 b29 b28<br>0 0 0 0: No data byte<br>0 0 0 1: 1 data byte<br>0 0 1 0: 2 data bytes<br>0 0 1 1: 3 data bytes<br>0 1 0 0: 4 data bytes<br>0 1 0 1: 5 data bytes<br>0 1 1 0: 6 data bytes<br>0 1 1 1: 7 data bytes<br>1 X X X: 8 data bytes |
| 27 to 16 | RMPTR[11:0] | Receive Buffer Label Data<br>Label information of the received message. |
| 15 to 0 | RMTS[15:0] | Receive Buffer Timestamp Data<br>Timestamp value of the received message. |

#### RMDLC[3:0] Bits

These bits indicate the data length of the message stored in the receive buffer.

#### RMPTR[11:0] Bits

These bits indicate the label information of the message stored in the receive buffer.

#### RMTS[15:0] Bits

These bits indicate the timestamp value of the message stored in the receive buffer.

### 21.4.5.5 RSCANnRMDF0q — Receive Buffer Data Field 0 Register (q = 0 to 95)

**Access:** RSCANnRMDF0q register is a read-only register that can be read in 32-bit units
RSCANnRMDF0qL, RSCANnRMDF0qH registers are read-only registers that can be read in 16-bit units
RSCANnRMDF0qLL, RSCANnRMDF0qLH, RSCANnRMDF0qHL, RSCANnRMDF0qHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnRMDF0q: <RSCANn_base> + $0608_H$ + ($10_H$ × q)

RSCANnRMDF0qL: <RSCANn_base> + $0608_H$ + ($10_H$ × q),
RSCANnRMDF0qH: <RSCANn_base> + $060A_H$ + ($10_H$ × q)

RSCANnRMDF0qLL: <RSCANn_base> + $0608_H$ + ($10_H$ × q),
RSCANnRMDF0qLH: <RSCANn_base> + $0609_H$ + ($10_H$ × q),
RSCANnRMDF0qHL: <RSCANn_base> + $060A_H$ + ($10_H$ × q),
RSCANnRMDF0qHH: <RSCANn_base> + $060B_H$ + ($10_H$ × q)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMDB3[7:0] | | | | | | | | RMDB2[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMDB1[7:0] | | | | | | | | RMDB0[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.41 RSCANnRMDF0q Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RMDB3[7:0] | Receive Buffer Data Byte 3 |
| 23 to 16 | RMDB2[7:0] | Receive Buffer Data Byte 2 |
| 15 to 8 | RMDB1[7:0] | Receive Buffer Data Byte 1 |
| 7 to 0 | RMDB0[7:0] | Receive Buffer Data Byte 0<br>Data for a message stored in the receive buffer can be read. |

When the RMDLC[3:0] value in the RSCANnRMPTRq register is smaller than $1000_B$, data bytes for which no data is set are read as $00_H$.

### 21.4.5.6 RSCANnRMDF1q — Receive Buffer Data Field 1 Register (q = 0 to 95)

**Access:** RSCANnRMDF1q register is a read-only register that can be read in 32-bit units
RSCANnRMDF1qL, RSCANnRMDF1qH registers are read-only registers that can be read in 16-bit units
RSCANnRMDF1qLL, RSCANnRMDF1qLH, RSCANnRMDF1qHL, RSCANnRMDF1qHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnRMDF1q: <RSCANn_base> + $060C_H$ + ($10_H$ × q)

RSCANnRMDF1qL: <RSCANn_base> + $060C_H$ + ($10_H$ × q),
RSCANnRMDF1qH: <RSCANn_base> + $060E_H$ + ($10_H$ × q)

RSCANnRMDF1qLL: <RSCANn_base> + $060C_H$ + ($10_H$ × q),
RSCANnRMDF1qLH: <RSCANn_base> + $060D_H$ + ($10_H$ × q),
RSCANnRMDF1qHL: <RSCANn_base> + $060E_H$ + ($10_H$ × q),
RSCANnRMDF1qHH: <RSCANn_base> + $060F_H$ + ($10_H$ × q)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMDB7[7:0] | | | | | | | | RMDB6[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMDB5[7:0] | | | | | | | | RMDB4[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.42 RSCANnRMDF1q Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RMDB7[7:0] | Receive Buffer Data Byte 7 |
| 23 to 16 | RMDB6[7:0] | Receive Buffer Data Byte 6 |
| 15 to 8 | RMDB5[7:0] | Receive Buffer Data Byte 5 |
| 7 to 0 | RMDB4[7:0] | Receive Buffer Data Byte 4<br>Data for a message stored in the receive buffer can be read. |

When the RMDLC[3:0] value in the RSCANnRMPTRq register is smaller than $1000_B$, data bytes for which no data is set are read as $00_H$.

## 21.4.6 Details of Receive FIFO Buffer-Related Registers

### 21.4.6.1 RSCANnRFCCx — Receive FIFO Buffer Configuration and Control Register (x = 0 to 7)

**Access:** RSCANnRFCCx register can be read or written in 32-bit units
RSCANnRFCCxL register can be read or written in 16-bit units
RSCANnRFCCxLL, RSCANnRFCCxLH registers can be read or written in 8-bit units

**Address:** RSCANnRFCCx: <RSCANn_base> + $00B8_H$ + ($04_H$ × x)

RSCANnRFCCxL: <RSCANn_base> + $00B8_H$ + ($04_H$ × x)

RSCANnRFCCxLL: <RSCANn_base> + $00B8_H$ + ($04_H$ × x),
RSCANnRFCCxLH: <RSCANn_base> + $00B9_H$ + ($04_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFIGCV[2:0] | | | RFIM | — | RFDC[2:0] | | | — | — | — | — | — | — | RFIE | RFE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

**Table 21.43 RSCANnRFCCx Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 15 to 13 | RFIGCV[2:0] | Receive FIFO Interrupt Request Timing Select<br>b15 b14 b13<br>0 0 0: When FIFO is 1/8 full.<br>0 0 1: When FIFO is 2/8 full.<br>0 1 0: When FIFO is 3/8 full.<br>0 1 1: When FIFO is 4/8 full.<br>1 0 0: When FIFO is 5/8 full.<br>1 0 1: When FIFO is 6/8 full.<br>1 1 0: When FIFO is 7/8 full.<br>1 1 1: When FIFO is full. |
| 12 | RFIM | Receive FIFO Interrupt Source Select<br>0: An interrupt occurs when the condition set by the RFIGCV[2:0] bits is met.<br>1: An interrupt occurs each time a message has been received. |
| 11 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 10 to 8 | RFDC[2:0] | Receive FIFO Buffer Depth Configuration<br>b10 b9 b8<br>0 0 0: 0 messages<br>0 0 1: 4 messages<br>0 1 0: 8 messages<br>0 1 1: 16 messages<br>1 0 0: 32 messages<br>1 0 1: 48 messages<br>1 1 0: 64 messages<br>1 1 1: 128 messages |
| 7 to 2 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Table 21.43 RSCANnRFCCx Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 1 | RFIE | Receive FIFO Interrupt Enable<br>0: Receive FIFO interrupt is disabled.<br>1: Receive FIFO interrupt is enabled. |
| 0 | RFE | Receive FIFO Buffer Enable<br>0: No receive FIFO buffer is used.<br>1: Receive FIFO buffers are used. |

### RFIGCV[2:0] Bits

These bits are used to specify the number of received messages for generating a receive FIFO interrupt request when the RFIM bit is set to 0 with a fraction for the total number of buffers (the setting of RFDC[2:0]).

When the RFDC[2:0] bits are set to $001_B$ (4 messages), set the RFIGCV[2:0] bits to $001_B$, $011_B$, $101_B$, or $111_B$. Modify these bits only in global reset mode.

### RFIM Bit

This bit is used to select a FIFO interrupt source. Modify this bit only in global reset mode.

### RFDC[2:0] Bits

These bits are used to select the number of messages that can be stored in a single receive FIFO buffer. When these bits are set to $000_B$, no receive FIFO buffer should be used. Modify these bits only in global reset mode.

### RFIE Bit

Setting the RFIE bit to 1 enables receive FIFO interrupts. Modify this bit when the RFE bit set to 0 (no receive FIFO buffer is used).

### RFE Bit

Setting the RFE bit to 1 makes receive FIFO buffers available. Clearing this bit to 0 sets the RFEMP flag in the RSCANnRFSTSx register to 1 (buffer empty). Modify this bit in global operating mode or global test mode.

Set this bit to 1 with another instruction after the settings to all bits in the RSCANnRFCCx register have been done.

This bit is cleared to 0 in global reset mode.

### 21.4.6.2 RSCANnRFSTSx — Receive FIFO Buffer Status Register (x = 0 to 7)

**Access:** RSCANnRFSTSx register can be read or written in 32-bit units
RSCANnRFSTSxL register can be read or written in 16-bit units
RSCANnRFSTSxLL register can be read or written in 8-bit units
RSCANnRFSTSxLH register is a read-only register that can be read in 8-bit units

**Address:** RSCANnRFSTSx: <RSCANn_base> + $00D8_H$ + ($04_H$ × x)

RSCANnRFSTSxL: <RSCANn_base> + $00D8_H$ + ($04_H$ × x)

RSCANnRFSTSxLL: <RSCANn_base> + $00D8_H$ + ($04_H$ × x),
RSCANnRFSTSxLH: <RSCANn_base> + $00D9_H$ + ($04_H$ × x)

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFMC[7:0] | | | | | | | | — | — | — | — | RFIF | RFMLT | RFFLL | RFEMP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W*1 | R/W*1 | R | R |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.44 RSCANnRFSTSx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 15 to 8 | RFMC[7:0] | Receive FIFO Unread Message Counter<br>The number of unread messages stored in the receive FIFO buffer is displayed. |
| 7 to 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | RFIF | Receive FIFO Interrupt Request Flag<br>0: No receive FIFO interrupt request is present.<br>1: A receive FIFO interrupt request is present. |
| 2 | RFMLT | Receive FIFO Message Lost Flag<br>0: No receive FIFO message is lost.<br>1: A receive FIFO message is lost. |
| 1 | RFFLL | Receive FIFO Buffer Full Status Flag<br>0: The receive FIFO buffer is not full.<br>1: The receive FIFO buffer is full. |
| 0 | RFEMP | Receive FIFO Buffer Empty Status Flag<br>0: The receive FIFO buffer contains unread message.<br>1: The receive FIFO buffer contains no unread message (buffer empty). |

#### RFMC[7:0] Flag

This flag indicates the number of unread messages in the receive FIFO buffer. This flag becomes $00_H$ when the RFE bit in the RSCANnRFCCx register is set to 0.

### RFIF Flag

This flag is set to 1 when the receive FIFO interrupt request generation conditions set by the RFIGCV[2:0] bits and the RFIM bit in the RSCANnRFCCx register are met. This flag is cleared to 0 in global reset mode or by writing 0 to this flag. Modify this bit in global operating mode or global test mode.
To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared. When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

### RFMLT Flag

This flag is set to 1 when an attempt is made to store a new message while the receive FIFO buffer is full. In this case, the new message is discarded.

This flag is cleared to 0 in global reset mode or by writing 0 to this flag.

Modify this bit in global operating mode or global test mode.

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared. When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

### RFFLL Flag

This flag is set to 1 when the number of messages stored in the receive FIFO buffer matches the FIFO buffer depth set by the RFDC[2:0] bits in the RSCANnRFCCx register.

If the number of messages stored in the receive FIFO buffer becomes smaller than the FIFO buffer depth set by the RFDC[2:0] bits, this flag is cleared to 0. This flag is also cleared to 0 when the RFE bit in the RSCANnRFCCx register is set to 0 (no receive FIFO buffer is used) or in global reset mode.

### RFEMP Flag

This flag is set to 1 when all messages in the receive FIFO buffer have been read. This flag is also set to 1 when the RFE bit in the RSCANnRFCCx register is 0 or in global reset mode.

This flag is cleared to 0 when even a single received message has been stored in the receive FIFO buffer.

**NOTE**

To clear the RFMLT or RFIF flag to 0, use a store instruction to write “0” to the given flag and “1” to the other flags.

### 21.4.6.3 RSCANnRFPCTRx — Receive FIFO Buffer Pointer Control Register (x = 0 to 7)

**Access:** RSCANnRFPCTRx register is a write-only register that can be written in 32-bit units
RSCANnRFPCTRxL register is a write-only register that can be written in 16-bit units
RSCANnRFPCTRxLL register is a write-only register that can be written in 8-bit units

**Address:** RSCANnRFPCTRx: <RSCANn_base> + $00F8_H$ + ($04_H$ × x)

RSCANnRFPCTRxL: <RSCANn_base> + $00F8_H$ + ($04_H$ × x)

RSCANnRFPCTRxLL: <RSCANn_base> + $00F8_H$ + ($04_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RFPC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**Table 21.45 RSCANnRFPCTRx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing, write the value after reset. |
| 7 to 0 | RFPC[7:0] | Receive FIFO Pointer Control<br>When these bits are set to $FF_H$, the read pointer moves to the next unread message in the receive FIFO buffer. |

#### RFPC[7:0] Bits

When the RFPC[7:0] bits are set to $FF_H$, the read pointer moves to the next unread message in the receive FIFO buffer. At this time, the RFMC[7:0] (receive FIFO unread message counter) value in the RSCANnRFSTSx register is decremented. Read the RSCANnRFIDx, RSCANnRFPTRx, RSCANnRFDF0x, and RSCANnRFDF1x registers to read messages in the receive FIFO buffer, and then write $FF_H$ to the RFPC[7:0] bits.

When writing $FF_H$ to these bits, make sure that the RFE bit in the RSCANnRFCCx register is set to 1 (receive FIFO buffers are used) and the RFEMP flag in the RSCANnRFSTSx register is 0 (the receive FIFO buffer contains unread messages).

### 21.4.6.4 RSCANnRFIDx — Receive FIFO Buffer Access ID Register (x = 0 to 7)

**Access:** RSCANnRFIDx register is a read-only register that can be read in 32-bit units
RSCANnRFIDxL, RSCANnRFIDxH registers are read-only registers that can be read in 16-bit units
RSCANnRFIDxLL, RSCANnRFIDxLH, RSCANnRFIDxHL, RSCANnRFIDxHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnRFIDx: <RSCANn_base> + $0E00_H$ + ($10_H$ × x)

RSCANnRFIDxL: <RSCANn_base> + $0E00_H$ + ($10_H$ × x),
RSCANnRFIDxH: <RSCANn_base> + $0E02_H$ + ($10_H$ × x)

RSCANnRFIDxLL: <RSCANn_base> + $0E00_H$ + ($10_H$ × x),
RSCANnRFIDxLH: <RSCANn_base> + $0E01_H$ + ($10_H$ × x),
RSCANnRFIDxHL: <RSCANn_base> + $0E02_H$ + ($10_H$ × x),
RSCANnRFIDxHH: <RSCANn_base> + $0E03_H$ + ($10_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFIDE | RFRTR | — | RFID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.46 RSCANnRFIDx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | RFIDE | Receive FIFO Buffer IDE<br>0: Standard ID<br>1: Extended ID |
| 30 | RFRTR | Receive FIFO Buffer RTR<br>0: Data frame<br>1: Remote frame |
| 29 | Reserved | When read, the value after reset is returned. |
| 28 to 0 | RFID[28:0] | Receive FIFO Buffer ID Data<br>The standard ID or extended ID of received message can be read.<br>Read bits b10 to b0 for standard ID. Bits b28 to b11 are read as 0. |

#### RFIDE Bit

This bit indicates the ID format (standard ID or extended ID) of the message stored in the receive FIFO buffer.

#### RFRTR Bit

This bit indicates the frame format (data frame or remote frame) of the message stored in the receive FIFO buffer.

#### RFID[28:0] Bits

These bits indicate the ID of the message stored in the receive FIFO buffer.

### 21.4.6.5 RSCANnRFPTRx — Receive FIFO Buffer Access Pointer Register (x = 0 to 7)

**Access:** RSCANnRFPTRx register is a read-only register that can be read in 32-bit units
RSCANnRFPTRxL, RSCANnRFPTRxH registers are read-only registers that can be read in 16-bit units
RSCANnRFPTRxLL, RSCANnRFPTRxLH, RSCANnRFPTRxHL, RSCANnRFPTRxHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnRFPTRx: <RSCANn_base> + $0E04_H$ + ($10_H$ × x)

RSCANnRFPTRxL: <RSCANn_base> + $0E04_H$ + ($10_H$ × x),
RSCANnRFPTRxH: <RSCANn_base> + $0E06_H$ + ($10_H$ × x)

RSCANnRFPTRxLL: <RSCANn_base> + $0E04_H$ + ($10_H$ × x),
RSCANnRFPTRxLH: <RSCANn_base> + $0E05_H$ + ($10_H$ × x),
RSCANnRFPTRxHL: <RSCANn_base> + $0E06_H$ + ($10_H$ × x),
RSCANnRFPTRxHH: <RSCANn_base> + $0E07_H$ + ($10_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFDLC[3:0] | | | | RFPTR[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFTS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.47 RSCANnRFPTRx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | RFDLC[3:0] | Receive FIFO Buffer DLC Data<br>b31 b30 b29 b28<br>0 0 0 0: 0 data bytes<br>0 0 0 1: 1 data byte<br>0 0 1 0: 2 data bytes<br>0 0 1 1: 3 data bytes<br>0 1 0 0: 4 data bytes<br>0 1 0 1: 5 data bytes<br>0 1 1 0: 6 data bytes<br>0 1 1 1: 7 data bytes<br>1 X X X: 8 data bytes |
| 27 to 16 | RFPTR[11:0] | Receive FIFO Buffer Label Data<br>Label information of the received message can be read. |
| 15 to 0 | RFTS[15:0] | Receive FIFO Buffer Timestamp Data<br>Timestamp value of the received message can be read. |

#### RFDLC[3:0] Bits

These bits contain the data length of the message stored in the receive FIFO buffer.

#### RFPTR[11:0] Bits

These bits contain the label information of the message stored in the receive FIFO buffer.

#### RFTS[15:0] Bits

These bits contain the timestamp value of the message stored in the receive FIFO buffer.

### 21.4.6.6 RSCANnRFDF0x — Receive FIFO Buffer Access Data Field 0 Register (x = 0 to 7)

**Access:** RSCANnRFDF0x register is a read-only register that can be read in 32-bit units
RSCANnRFDF0xL, RSCANnRFDF0xH registers are read-only registers that can be read in 16-bit units
RSCANnRFDF0xLL, RSCANnRFDF0xLH, RSCANnRFDF0xHL, RSCANnRFDF0xHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnRFDF0x: <RSCANn_base> + $0E08_H$ + ($10_H$ × x)

RSCANnRFDF0xL: <RSCANn_base> + $0E08_H$ + ($10_H$ × x),
RSCANnRFDF0xH: <RSCANn_base> + $0E0A_H$ + ($10_H$ × x)

RSCANnRFDF0xLL: <RSCANn_base> + $0E08_H$ + ($10_H$ × x),
RSCANnRFDF0xLH: <RSCANn_base> + $0E09_H$ + ($10_H$ × x),
RSCANnRFDF0xHL: <RSCANn_base> + $0E0A_H$ + ($10_H$ × x),
RSCANnRFDF0xHH: <RSCANn_base> + $0E0B_H$ + ($10_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFDB3[7:0] | | | | | | | | RFDB2[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFDB1[7:0] | | | | | | | | RFDB0[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.48 RSCANnRFDF0x Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RFDB3[7:0] | Receive FIFO Buffer Data Byte 3 |
| 23 to 16 | RFDB2[7:0] | Receive FIFO Buffer Data Byte 2 |
| 15 to 8 | RFDB1[7:0] | Receive FIFO Buffer Data Byte 1 |
| 7 to 0 | RFDB0[7:0] | Receive FIFO Buffer Data Byte 0<br>Data for a message stored in the receive FIFO buffer can be read. |

When the RFDLC[3:0] value in the RSCANnRFPTRx register is smaller than $1000_B$, data bytes for which no data is set are read as $00_H$.

### 21.4.6.7 RSCANnRFDF1x — Receive FIFO Buffer Access Data Field 1 Register (x = 0 to 7)

**Access:** RSCANnRFDF1x register is a read-only register that can be read in 32-bit units
RSCANnRFDF1xL, RSCANnRFDF1xH registers are read-only registers that can be read in 16-bit units
RSCANnRFDF1xLL, RSCANnRFDF1xLH, RSCANnRFDF1xHL, RSCANnRFDF1xHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnRFDF1x: <RSCANn_base> + $0E0C_H$ + ($10_H$ × x)

RSCANnRFDF1xL: <RSCANn_base> + $0E0C_H$ + ($10_H$ × x),
RSCANnRFDF1xH: <RSCANn_base> + $0E0E_H$ + ($10_H$ × x)

RSCANnRFDF1xLL: <RSCANn_base> + $0E0C_H$ + ($10_H$ × x),
RSCANnRFDF1xLH: <RSCANn_base> + $0E0D_H$ + ($10_H$ × x),
RSCANnRFDF1xHL: <RSCANn_base> + $0E0E_H$ + ($10_H$ × x),
RSCANnRFDF1xHH: <RSCANn_base> + $0E0F_H$ + ($10_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFDB7[7:0] | | | | | | | | RFDB6[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFDB5[7:0] | | | | | | | | RFDB4[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.49 RSCANnRFDF1x Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RFDB7[7:0] | Receive FIFO Buffer Data Byte 7 |
| 23 to 16 | RFDB6[7:0] | Receive FIFO Buffer Data Byte 6 |
| 15 to 8 | RFDB5[7:0] | Receive FIFO Buffer Data Byte 5 |
| 7 to 0 | RFDB4[7:0] | Receive FIFO Buffer Data Byte 4<br>Data for a message stored in the receive FIFO buffer can be read. |

When the RFDLC[3:0] value in the RSCANnRFPTRx register is smaller than $1000_B$, data bytes for which no data is set are read as $00_H$.

## 21.4.7 Details of Transmit/Receive FIFO Buffer-Related Registers

### 21.4.7.1 RSCANnCFCCk — Transmit/receive FIFO Buffer Configuration and Control Register k (k = 0 to 17)

**Access:** RSCANnCFCCk register can be read or written in 32-bit units
RSCANnCFCCkL, RSCANnCFCCkH registers can be read or written in 16-bit units
RSCANnCFCCkLL, RSCANnCFCCkLH, RSCANnCFCCkHL, RSCANnCFCCkHH registers can be read or written in 8-bit units

**Address:** RSCANnCFCCk: <RSCANn_base> + $0118_H$ + ($04_H$ × k)

RSCANnCFCCkL: <RSCANn_base> + $0118_H$ + ($04_H$ × k),
RSCANnCFCCkH: <RSCANn_base> + $011A_H$ + ($04_H$ × k)

RSCANnCFCCkLL: <RSCANn_base> + $0118_H$ + ($04_H$ × k),
RSCANnCFCCkLH: <RSCANn_base> + $0119_H$ + ($04_H$ × k),
RSCANnCFCCkHL: <RSCANn_base> + $011A_H$ + ($04_H$ × k),
RSCANnCFCCkHH: <RSCANn_base> + $011B_H$ + ($04_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFITT[7:0] | | | | | | | | CFTML[3:0] | | | | CFITR | CFITSS | CFM[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFIGCV[2:0] | | | CFIM | — | CFDC[2:0] | | | — | — | — | — | — | CFTXIE | CFRXIE | CFE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

**Table 21.50 RSCANnCFCCk Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | CFITT[7:0] | Set a message transmission interval.<br>Set Value: $00_H$ to $FF_H$ |
| 23 to 20 | CFTML[3:0] | Transmit Buffer Link Configuration<br>Set the transmit buffer number to be linked to the transmit/receive FIFO buffer. |
| 19 | CFITR | Transmit/Receive FIFO Interval Timer Resolution<br>0: Clock dividing pclk/2 by (ITRCP[15:0] bits)<br>1: Clock dividing pclk/2 by (ITRCP[15:0] bits × 10) |
| 18 | CFITSS | Transmit/Receive FIFO Interval Timer Clock Source Select<br>0: Interval timer clock source selected by the CFITR bit<br>1: Interval timer clock source is the bit time clock for the channel to which the FIFO is linked. |
| 17, 16 | CFM[1:0] | Transmit/Receive FIFO Mode Select<br>b17 b16<br>0 0: Receive mode<br>0 1: Transmit mode<br>1 0: Gateway mode<br>1 1: Setting prohibited |
| 15 to 13 | CFIGCV[2:0] | Transmit/Receive FIFO Receive Interrupt Request Timing Select<br>b15 b14 b13<br>0 0 0: When FIFO is 1/8 full.<br>0 0 1: When FIFO is 2/8 full.<br>0 1 0: When FIFO is 3/8 full.<br>0 1 1: When FIFO is 4/8 full.<br>1 0 0: When FIFO is 5/8 full.<br>1 0 1: When FIFO is 6/8 full.<br>1 1 0: When FIFO is 7/8 full.<br>1 1 1: When FIFO is full. |

Table 21.50 RSCANnCFCCk Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 12 | CFIM | Transmit/Receive FIFO Interrupt Source Select<br>0:<br>• Receive mode/gateway mode<br>When the number of received messages has met the condition set by the CFIGCV[2:0] bits, a FIFO receive interrupt request is generated.<br>• Transmit mode/gateway mode<br>When the buffer becomes empty upon completion of message transmission, a FIFO transmit interrupt request is generated.<br>1:<br>• Receive mode/gateway mode<br>A FIFO receive interrupt request is generated each time a message has been received.<br>• Transmit mode/gateway mode<br>A FIFO transmit interrupt request is generated each time a message has been transmitted. |
| 11 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 10 to 8 | CFDC[2:0] | Transmit/Receive FIFO Buffer Depth Configuration<br>b10 b9 b8<br>0 0 0: 0 messages<br>0 0 1: 4 messages<br>0 1 0: 8 messages<br>0 1 1: 16 messages<br>1 0 0: 32 messages<br>1 0 1: 48 messages<br>1 1 0: 64 messages<br>1 1 1: 128 messages |
| 7 to 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 | CFTXIE | Transmit/Receive FIFO Transmit Interrupt Enable<br>0: Transmit/receive FIFO transmit interrupt is disabled.<br>1: Transmit/receive FIFO transmit interrupt is enabled. |
| 1 | CFRXIE | Transmit/Receive FIFO Receive Interrupt Enable<br>0: Transmit/receive FIFO receive interrupt is disabled.<br>1: Transmit/receive FIFO receive interrupt is enabled. |
| 0 | CFE | Transmit/Receive FIFO Buffer Enable<br>0: No transmit/receive FIFO buffer is used.<br>1: Transmit/receive FIFO buffers are used. |

### CFITT[7:0] Bits

These bits are used to set a message transmission interval when transmitting messages continuously from a transmit/receive FIFO buffer whose CFM[1:0] bits are set to $01_B$ (transmit mode) or $10_B$ (gateway mode).

Clear the CFE bit to 0 (no transmit/receive FIFO buffer is used) before modifying the CFITT[7:0] bits.

### CFTML[3:0] Bits

These bits are used to set the number of transmit buffer on the channel which will be linked to transmit/receive FIFO buffer k when the CFM[1:0] bits are set to $01_B$ (transmit mode) or $10_B$ (gateway mode). There are three transmit/receive FIFO buffers per channel, so channel number m of FIFO buffer k is calculated as m = k/3 (integer division). The actual assigned transmit buffer number p linked to FIFO buffer k will be ((16 × m) + CFTML[3:0]).

See **Table 21.15** and **Table 21.16**, for the relationship between transmit/receive FIFO buffer k and transmit buffer p.

Setting the CFDC[2:0] bits to $001_B$ or more enables the setting of the CFTML[3:0] bits.

Do not link to any transmit buffer which is already allocated to a transmit queue on the identical channel or to another transmit/receive FIFO buffer. Modify these bits only in global reset mode.

#### CFITR Bit

This bit is enabled when the CFITSS bit is 0.

When this bit is 0, the interval timer clock source is the pclk/2 clock divided by the value of the ITRCP[15:0] bits in the RSCANnGCFG register.

When this bit is 1, the interval timer clock source is the pclk/2 clock divided by the value of the ITRCP[15:0] bits in the RSCANnGCFG register × 10.

Modify this bit while the CFE bit is set to 0 (no transmit/receive FIFO buffer is used).

#### CFITSS Bit

When this bit is 0, the clock selected by the CFITR bit is the count source of the interval timer.

When this bit is 1, the bit time clock of the channel to which the FIFO is linked is the count source of the interval timer.

Modify this bit while the CFE bit is set to 0 (no transmit/receive FIFO buffer is used).

#### CFM[1:0] Bits

These bits are used to select transmit/receive FIFO mode. Modify these bits only in global reset mode.

#### CFIGCV[2:0] Bits

These bits are used to specify the number of received messages for generating a transmit/receive FIFO receive interrupt request when the CFM[1:0] bits are set to $00_B$ (receive mode) or $10_B$ (gateway mode) and the CFIM bit is set to 0 with a fraction for the total number of buffers (the setting of CFDC[2:0]).

When the CFDC[2:0] bits are set to $001_B$ (4 messages), set the CFIGCV[2:0] bits to $001_B$, $011_B$, $101_B$, or $111_B$.

Modify these bits only in global reset mode.

#### CFIM Bit

This bit is used to select a transmit/receive FIFO interrupt source. Modify this bit only in global reset mode.

#### CFDC[2:0] Bits

These bits are used to set the number of messages that can be stored in a single transmit/receive FIFO buffer. When these bits are set to $000_B$, do not use a transmit/receive FIFO buffer. Modify these bits only in global reset mode.

#### CFTXIE Bit

When this bit is set to 1 and the CFTXIF flag in the RSCANnCFSTSk register is set to 1, a transmit/receive FIFO transmit interrupt request is generated.

Modify this bit with the CFE bit set to 0 (no transmit/receive FIFO buffer is used).

### CFRXIE Bit

When this bit is set to 1 and the CFRXIF flag in the RSCANnCFSTSk register is set to 1, a transmit/receive FIFO receive interrupt request is generated.

Modify this bit with the CFE bit set to 0.

### CFE Bit

Setting this bit to 1 makes transmit/receive FIFO buffers available.

When this bit is set to 0 in transmit mode or gateway mode, if a message in the transmit/receive FIFO buffer is being transmitted or will be transmitted next, the transmit/receive FIFO buffer becomes empty after completion of transmission of that message, or upon detection of a CAN bus error, or arbitration-lost. In other cases or in receive mode, the transmit/receive FIFO buffer becomes empty immediately.

This bit is cleared to 0 when the following conditions are met.

- Receive mode: Global reset mode
- Transmit mode or gateway mode: Channel reset mode

Modify this bit in the following mode.

- Receive mode: Global operating mode or global test mode
- Transmit mode or gateway mode: Channel communication mode or channel halt mode

After all other bits in the RSCANnCFCCk register have been set, set this bit to 1 by using another instruction.

### 21.4.7.2 RSCANnCFSTSk — Transmit/receive FIFO Buffer Status Register (k = 0 to 17)

**Access:** RSCANnCFSTSk register can be read or written in 32-bit units
RSCANnCFSTSkL register can be read or written in 16-bit units
RSCANnCFSTSkLL register can be read or written in 8-bit units
RSCANnCFSTSkLH register is a read-only register that can be read in 8-bit units

**Address:** RSCANnCFSTSk: <RSCANn_base> + $0178_H$ + ($04_H$ × k)

RSCANnCFSTSkL: <RSCANn_base> + $0178_H$ + ($04_H$ × k)

RSCANnCFSTSkLL: <RSCANn_base> + $0178_H$ + ($04_H$ × k),
RSCANnCFSTSkLH: <RSCANn_base> + $0179_H$ + ($04_H$ × k)

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFMC[7:0] | | | | | | | | — | — | — | CFTXIF | CFRXIF | CFMLT | CFFLL | CFEMP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W*1 | R/W*1 | R/W*1 | R | R |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.51 RSCANnCFSTSk Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 15 to 8 | CFMC[7:0] | Transmit/Receive FIFO Message Counter<br>The number of messages stored in the transmit/receive FIFO buffer. |
| 7 to 5 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 4 | CFTXIF | Transmit/Receive FIFO Transmit Interrupt Request Flag<br>0: No transmit/receive FIFO transmit interrupt request is present.<br>1: A transmit/receive FIFO transmit interrupt request is present. |
| 3 | CFRXIF | Transmit/Receive FIFO Receive Interrupt Request Flag<br>0: No transmit/receive FIFO receive interrupt request is present.<br>1: A transmit/receive FIFO receive interrupt request is present. |
| 2 | CFMLT | Transmit/Receive FIFO Message Lost Flag<br>0: No transmit/receive FIFO message is lost.<br>1: A transmit/receive FIFO message is lost. |
| 1 | CFFLL | Transmit/Receive FIFO Buffer Full Status Flag<br>0: The transmit/receive FIFO buffer is not full.<br>1: The transmit/receive FIFO buffer is full. |
| 0 | CFEMP | Transmit/Receive FIFO Buffer Empty Status Flag<br>0: The transmit/receive FIFO buffer contains messages.<br>1: The transmit/receive FIFO buffer contains no message (buffer empty). |

#### CFMC[7:0] Bits

The CFMC[7:0] bits indicate the following values that depend on the setting of the CFM[1:0] bits in the RSCANnCFCCk register.

- When CFM[1:0] value is $01_B$ (transmit mode): Number of untransmitted messages in the buffer
- When CFM[1:0] value is $00_B$ (receive mode): Number of unread received messages in the buffer

- When CFM[1:0] value is $10_B$ (gateway mode): Number of untransmitted received messages in the buffer

These bits are cleared to 0 when any of the following conditions is met.

- When CFM[1:0] value is $00_B$: In global reset mode
- When CFM[1:0] value is $01_B$ or $10_B$: In channel reset mode
- When the CFE bit in the RSCANnCFCCk register is cleared to 0.

#### CFTXIF Flag

The CFTXIF flag is set to 1 when any of the following conditions is met.

- When the CFM[1:0] bits are set to $01_B$ or $10_B$, and the factor selected by the CFIM bit in the RSCANnCFCCk register occurs

The CFTXIF flag is cleared to 0 when any of the following conditions is met.

- When 0 is written to the CFTXIF flag
- When the CFM[1:0] bits are set to $00_B$: In global reset mode
- When the CFM[1:0] bits are set to $01_B$ or $10_B$: In channel reset mode

Write 0 to this flag in global operating mode or global test mode.

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared. When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

#### CFRXIF Flag

The CFRXIF flag is set to 1 when any of the following conditions is met.

- When the CFM[1:0] bits are set to $00_B$ or $10_B$, and the factor selected by the CFIM bit in the RSCANnCFCCk register occurs

The CFRXIF flag is cleared to 0 when any of the following conditions is met.

- When 0 is written to the CFRXIF flag
- When the CFM[1:0] bits are set to $00_B$: In global reset mode
- When the CFM[1:0] bits are set to $01_B$ or $10_B$: In channel reset mode

Write 0 to this flag in global operating mode or global test mode.

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared.

When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

#### CFMLT Flag

The CFMLT flag is set to 1 when any of the following conditions is met.

- When an attempt is made to store a new message while the transmit/receive FIFO buffer is full. In this case, the new message is discarded.

The CFMLT flag is cleared to 0 when any of the following conditions is met.

- When 0 is written to the CFMLT flag

- When the CFM[1:0] bits are set to $00_B$: In global reset mode
- When the CFM[1:0] bits are set to $01_B$ or $10_B$: In channel reset mode

Write 0 to this flag in global operating mode or global test mode

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared. When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

#### CFFLL Flag

The CFFLL flag is set to 1 when any of the following conditions is met.

- When the number of messages stored in the transmit/receive FIFO buffer matches the FIFO buffer depth set by the CFDC[2:0] bits in the RSCANnCFCCk register.

The CFMLT flag is cleared to 0 when any of the following conditions is met.

- When the number of messages stored in the transmit/receive FIFO buffer becomes smaller than the FIFO buffer depth set by the CFDC[2:0] bits.
- When the CFE bit in the RSCANnCFCCk register is 0 (no transmit/receive FIFO buffer is used): When not performing the transmit abort
- When the CFM[1:0] bits are set to $00_B$: In global reset mode
- When the CFM[1:0] bits are set to $01_B$ or $10_B$: In channel reset mode

#### CFEMP Flag

The CFEMP flag is set to 1 when any of the following conditions is met.

- When the CFM[1:0] bits are set to $00_B$: All messages have been read, or in global reset mode
- When the CFM[1:0] bits are set to $01_B$ or $10_B$: All messages have been transmitted, or in channel reset mode
- When the CFE bit is 0 (no transmit/receive FIFO buffer is used): Not in the transmit abort

The CFEMP flag is cleared to 0 when any of the following conditions is met.

- When the CFM[1:0] bits are set to $00_B$ or $10_B$: At least one received message has been stored in the transmit/receive FIFO buffer.
- When the CFM[1:0] bits are set to $01_B$: A value of $FF_H$ has been written to the RSCANnCFPCTRk register after data was written to the RSCANnCFIDk, RSCANnCFPTRk, RSCANnCFDF0k, and RSCANnCFDF1k registers.

**NOTE**

To clear CFTXIF, CFRXIF, or CFMLT flag to 0, the program must write 0. When writing, use a store instruction to write “0” to the given flag and “1” to other flags.

### 21.4.7.3 RSCANnCFPCTRk — Transmit/receive FIFO Buffer Pointer Control Register (k = 0 to 17)

**Access:** RSCANnCFPCTRk register is a write-only register that can be written in 32-bit units
RSCANnCFPCTRkL register is a write-only register that can be written in 16-bit units
RSCANnCFPCTRkLL register is a write-only register that can be written in 8-bit units

**Address:** RSCANnCFPCTRk: <RSCANn_base> + $01D8_H$ + ($04_H$ × k)

RSCANnCFPCTRkL: <RSCANn_base> + $01D8_H$ + ($04_H$ × k)

RSCANnCFPCTRkLL: <RSCANn_base> + $01D8_H$ + ($04_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CFPC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**Table 21.52 RSCANnCFPCTRk Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing, write the value after reset. |
| 7 to 0 | CFPC[7:0] | Transmit/Receive FIFO Pointer Control<br>• Receive mode:<br>Writing $FF_H$ to these bits moves the read pointer to the next unread message in the transmit/receive FIFO buffer.<br>• Transmit mode:<br>Writing $FF_H$ to these bits moves the write pointer to the next stage of the transmit/receive FIFO buffer.<br>• Gateway mode:<br>Setting prohibited |

#### CFPC[7:0] Bits

- Receive mode (CFM[1:0] value in the RSCANnCFCCk register is $00_B$):
  Writing $FF_H$ to the CFPC[7:0] bits moves the read pointer to the next unread message in the transmit/receive FIFO buffer. At this time, the CFMC[7:0] value (transmit/receive FIFO message counter) in the RSCANnCFSTSk register is decremented. Read the RSCANnCFIDk, RSCANnCFPTRk, RSCANnCFDF0k, and RSCANnCFDF1k registers to read messages from the transmit/receive FIFO buffer, and then write $FF_H$ to the CFPC[7:0] bits.
  When writing $FF_H$ to these bits, make sure that the CFE bit in the RSCANnCFCCk register is set to 1 (transmit/receive FIFO buffers are used) and the CFEMP flag in the RSCANnCFSTSk register is 0 (the transmit/receive FIFO buffer contains messages).

- Transmit mode (CFM[1:0] value in the RSCANnCFCCk register is $01_B$):
  Writing $FF_H$ to the CFPC[7:0] bits stores the data written to the RSCANnCFIDk, RSCANnCFPTRk, RSCANnCFDF0k, and RSCANnCFDF1k registers in the transmit/receive FIFO buffer and moves the write pointer to the next stage of the transmit/receive FIFO buffer. At this time, the CFMC[7:0] value is incremented. Write transmit messages to the RSCANnCFIDk, RSCANnCFPTRk, RSCANnCFDF0k, and RSCANnCFDF1k registers before writing $FF_H$ to the CFPC[7:0] bits.
  When writing $FF_H$ to these bits, make sure that the CFE bit in the RSCANnCFCCk register is set to 1 and the CFFLL flag in the RSCANnCFSTSk register is 0 (the transmit/receive FIFO buffer is not full).
- Gateway mode (CFM[1:0] value in the RSCANnCFCCk register is $10_B$):
  Setting prohibited

### 21.4.7.4 RSCANnCFIDk — Transmit/receive FIFO Buffer Access ID Register (k = 0 to 17)

**Access:** RSCANnCFIDk register can be read or written in 32-bit units
RSCANnCFIDkL, RSCANnCFIDkH registers can be read or written in 16-bit units
RSCANnCFIDkLL, RSCANnCFIDkLH, RSCANnCFIDkHL, RSCANnCFIDkHH registers can be read or written in 8-bit units

**Address:** RSCANnCFIDk: <RSCANn_base> + $0E80_H$ + ($10_H$ × k)

RSCANnCFIDkL: <RSCANn_base> + $0E80_H$ + ($10_H$ × k),
RSCANnCFIDkH: <RSCANn_base> + $0E82_H$ + ($10_H$ × k)

RSCANnCFIDkLL: <RSCANn_base> + $0E80_H$ + ($10_H$ × k),
RSCANnCFIDkLH: <RSCANn_base> + $0E81_H$ + ($10_H$ × k),
RSCANnCFIDkHL: <RSCANn_base> + $0E82_H$ + ($10_H$ × k),
RSCANnCFIDkHH: <RSCANn_base> + $0E83_H$ + ($10_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFIDE | CFRTR | THLEN | CFID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.53 RSCANnCFIDk Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | CFIDE | Transmit/Receive FIFO Buffer IDE<br>0: Standard ID<br>1: Extended ID |
| 30 | CFRTR | Transmit/Receive FIFO Buffer RTR<br>0: Data frame<br>1: Remote frame |
| 29 | THLEN | Transmit History Data Store Enable<br>This bit is valid only when the CFM[1:0] value is $01_B$ (transmit mode).<br>0: Transmit history data is not stored in the buffer.<br>1: Transmit history data is stored in the buffer. |
| 28 to 0 | CFID[28:0] | Transmit/Receive FIFO Buffer ID Data<br>• When CFM[1:0] value is $01_B$ (transmit mode):<br>Set standard ID or extended ID. For standard ID, write an ID to bits 10 to 0 and write 0 to bits 28 to 11.<br>• When CFM[1:0] value is $00_B$ (receive mode):<br>Standard ID or extended ID in the received message can be read. For standard ID, read bits 10 to 0. Bits 28 to 11 are read as 0. |

This register is writable only when the CFM[1:0] value in the RSCANnCFCCk register is $01_B$ (transmit mode). This register is readable only when the CFM[1:0] value is $00_B$ (receive mode). This RSCANnCFIDk register should not be accessed when the CFM[1:0] value is $10_B$ (gateway mode).

#### CFIDE Bit

This bit indicates the ID format (standard ID or extended ID) of the received message stored in the transmit/receive FIFO buffer when the CFM[1:0] value is $00_B$. When the CFM[1:0] value is $01_B$, these bits are used to set the ID format of the message to be transmitted from the transmit/receive FIFO buffer.

#### CFRTR Bit

This bit indicates the data format (data frame or remote frame) of the received message stored in the transmit/receive FIFO buffer when the CFM[1:0] value is $00_B$. When the CFM[1:0] value is $01_B$, this bit is used to set the data format of the message to be transmitted from the transmit/receive FIFO buffer.

#### THLEN Bit

When this bit is set to 1, the transmit history data (label information, buffer number, and buffer type) of transmit messages is stored in the transmit history buffer after transmission is completed.

This bit is enabled when the CFM[1:0] value is $01_B$ (transmit mode).

#### CFID[28:0] Bits

These bits contain the ID of the received message stored in the transmit/receive FIFO buffer when the CFM[1:0] value is $00_B$.

When the CFM[1:0] value is $01_B$, this bit is used to set the ID of the message to be transmitted from the transmit/receive FIFO buffer.

### 21.4.7.5 RSCANnCFPTRk — Transmit/receive FIFO Buffer Access Pointer Register (k = 0 to 17)

**Access:** RSCANnCFPTRk register can be read or written in 32-bit units
RSCANnCFPTRkL, RSCANnCFPTRkH registers can be read or written in 16-bit units
RSCANnCFPTRkLL, RSCANnCFPTRkLH, RSCANnCFPTRkHL, RSCANnCFPTRkHH registers can be read or written in 8-bit units

**Address:** RSCANnCFPTRk: <RSCANn_base> + $0E84_H$ + ($10_H$ × k)

RSCANnCFPTRkL: <RSCANn_base> + $0E84_H$ + ($10_H$ × k),
RSCANnCFPTRkH: <RSCANn_base> + $0E86_H$ + ($10_H$ × k)

RSCANnCFPTRkLL: <RSCANn_base> + $0E84_H$ + ($10_H$ × k),
RSCANnCFPTRkLH: <RSCANn_base> + $0E85_H$ + ($10_H$ × k),
RSCANnCFPTRkHL: <RSCANn_base> + $0E86_H$ + ($10_H$ × k),
RSCANnCFPTRkHH: <RSCANn_base> + $0E87_H$ + ($10_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFDLC[3:0] | | | | CFPTR[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFTS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.54 RSCANnCFPTRk Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | CFDLC[3:0] | Transmit/Receive FIFO Buffer DLC Data<br>b31 b30 b29 b28<br>0 0 0 0: 0 data bytes<br>0 0 0 1: 1 data byte<br>0 0 1 0: 2 data bytes<br>0 0 1 1: 3 data bytes<br>0 1 0 0: 4 data bytes<br>0 1 0 1: 5 data bytes<br>0 1 1 0: 6 data bytes<br>0 1 1 1: 7 data bytes<br>1 X X X: 8 data bytes |
| 27 to 16 | CFPTR[11:0] | Transmit/Receive FIFO Buffer Label Data<br>• When CFM[1:0] value is $01_B$ (transmit mode):<br>Set the label information to be stored in the transmit history buffer.<br>Only bits CFPTR[7:0] are valid.<br>• When CFM[1:0] value is $00_B$ (receive mode):<br>The label information of the received message can be read. |
| 15 to 0 | CFTS[15:0] | Transmit/Receive FIFO Buffer Timestamp Data<br>These bits are valid only when the CFM[1:0] value is $00_B$ (receive mode).<br>The timestamp value of the received message can be read. |

This register is writable only when the CFM[1:0] value in the RSCANnCFCCk register is $01_B$ (transmit mode). This register is readable only when the CFM[1:0] value is $00_B$ (receive mode). This register should not be read or written when the CFM[1:0] value is $10_B$ (gateway mode).

#### CFDLC[3:0] Bits

These bits indicate the data length of the received message stored in the transmit/receive FIFO buffer when the CFM[1:0] value is $00_B$. When the CFM[1:0] value is $01_B$, these bits are used to set the data length of the message to be transmitted from the transmit/receive FIFO buffer. If the data length is set to $1001_B$ or more, the actual transmit data defaults to 8 bytes.

#### CFPTR[11:0] Bits

These bits indicate the label information attached to the received message stored in the transmit/receive FIFO buffer when the CFM[1:0] value is $00_B$. When the CFM[1:0] value is $01_B$, the CFPTR[7:0] value is stored in the transmit history buffer when message transmission has been completed.

#### CFTS[15:0] Bits

These bits indicate the timestamp value of the message stored in the transmit/receive FIFO buffer.

These bits are valid when the CFM[1:0] value is $00_B$.

### 21.4.7.6 RSCANnCFDF0k — Transmit/receive FIFO Buffer Access Data Field 0 Register (k = 0 to 17)

**Access:** RSCANnCFDF0k register can be read or written in 32-bit units
RSCANnCFDF0kL, RSCANnCFDF0kH registers can be read or written in 16-bit units
RSCANnCFDF0kLL, RSCANnCFDF0kLH, RSCANnCFDF0kHL, RSCANnCFDF0kHH registers can be read or written in 8-bit units

**Address:** RSCANnCFDF0k: <RSCANn_base> + $0E88_H$ + ($10_H$ × k)

RSCANnCFDF0kL: <RSCANn_base> + $0E88_H$ + ($10_H$ × k),
RSCANnCFDF0kH: <RSCANn_base> + $0E8A_H$ + ($10_H$ × k)

RSCANnCFDF0kLL: <RSCANn_base> + $0E88_H$ + ($10_H$ × k),
RSCANnCFDF0kLH: <RSCANn_base> + $0E89_H$ + ($10_H$ × k),
RSCANnCFDF0kHL: <RSCANn_base> + $0E8A_H$ + ($10_H$ × k),
RSCANnCFDF0kHH: <RSCANn_base> + $0E8B_H$ + ($10_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFDB3[7:0] | | | | | | | | CFDB2[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFDB1[7:0] | | | | | | | | CFDB0[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.55 RSCANnCFDF0k Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | CFDB3[7:0] | Transmit/Receive FIFO Buffer Data Byte 3<br>Transmit/Receive FIFO Buffer Data Byte 2<br>Transmit/Receive FIFO Buffer Data Byte 1<br>Transmit/Receive FIFO Buffer Data Byte 0<br>• When CFM[1:0] value is $01_B$ (transmit mode): Set the transmit/receive FIFO buffer data.<br>• When CFM[1:0] value is $00_B$ (receive mode): The message data stored in the transmit/receive FIFO buffer can be read. |
| 23 to 16 | CFDB2[7:0] | |
| 15 to 8 | CFDB1[7:0] | |
| 7 to 0 | CFDB0[7:0] | |

This register is writable only when the CFM[1:0] value in the RSCANnCFCCk register is $01_B$ (transmit mode).

This register is readable only when the CFM[1:0] value is $00_B$ (receive mode). When the CFDLC[3:0] value in the RSCANnCFPTRk register is smaller than $1000_B$, data bytes for which no data is set are read as $00_H$.

This register should not be read or written when the CFM[1:0] value is $10_B$ (gateway mode).

### 21.4.7.7 RSCANnCFDF1k — Transmit/receive FIFO Buffer Access Data Field 1 Register (k = 0 to 17)

**Access:** RSCANnCFDF1k register can be read or written in 32-bit units
RSCANnCFDF1kL, RSCANnCFDF1kH registers can be read or written in 16-bit units
RSCANnCFDF1kLL, RSCANnCFDF1kLH, RSCANnCFDF1kHL, RSCANnCFDF1kHH registers can be read or written in 8-bit units

**Address:** RSCANnCFDF1k: <RSCANn_base> + $0E8C_H$ + ($10_H$ × k)

RSCANnCFDF1kL: <RSCANn_base> + $0E8C_H$ + ($10_H$ × k),
RSCANnCFDF1kH: <RSCANn_base> + $0E8E_H$ + ($10_H$ × k)

RSCANnCFDF1kLL: <RSCANn_base> + $0E8C_H$ + ($10_H$ × k),
RSCANnCFDF1kLH: <RSCANn_base> + $0E8D_H$ + ($10_H$ × k),
RSCANnCFDF1kHL: <RSCANn_base> + $0E8E_H$ + ($10_H$ × k),
RSCANnCFDF1kHH: <RSCANn_base> + $0E8F_H$ + ($10_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFDB7[7:0] | | | | | | | | CFDB6[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFDB5[7:0] | | | | | | | | CFDB4[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.56 RSCANnCFDF1k Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | CFDB7[7:0] | Transmit/Receive FIFO Buffer Data Byte 7<br>Transmit/Receive FIFO Buffer Data Byte 6<br>Transmit/Receive FIFO Buffer Data Byte 5<br>Transmit/Receive FIFO Buffer Data Byte 4<br>• When CFM[1:0] value is $01_B$ (transmit mode): Set the transmit/receive FIFO buffer data.<br>• When CFM[1:0] value is $00_B$ (receive mode): The message data stored in the transmit/receive FIFO buffer can be read. |
| 23 to 16 | CFDB6[7:0] | |
| 15 to 8 | CFDB5[7:0] | |
| 7 to 0 | CFDB4[7:0] | |

This register is writable only when the CFM[1:0] value in the RSCANnCFCCk register is $01_B$ (transmit mode).

This register is readable only when the CFM[1:0] value is $00_B$ (receive mode). When the CFDLC[3:0] value in the RSCANnCFPTRk register is smaller than $1000_B$, data bytes for which no data is set are read as $00_H$.

This register should not be read or written when the CFM[1:0] value is $10_B$ (gateway mode).

## 21.4.8 Details of FIFO Status-Related Registers

### 21.4.8.1 RSCANnFESTS — FIFO Empty Status Register

**Access:** RSCANnFESTS register is a read-only register that can be read in 32-bit units
RSCANnFESTSL, RSCANnFESTSH registers are read-only registers that can be read in 16-bit units
RSCANnFESTSLL, RSCANnFESTSLH, RSCANnFESTSHL, RSCANnFESTSHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnFESTS: <RSCANn_base> + $0238_H$

RSCANnFESTSL: <RSCANn_base> + $0238_H$,
RSCANnFESTSH: <RSCANn_base> + $023A_H$

RSCANnFESTSLL: <RSCANn_base> + $0238_H$,
RSCANnFESTSLH: <RSCANn_base> + $0239_H$,
RSCANnFESTSHL: <RSCANn_base> + $023A_H$,
RSCANnFESTSHH: <RSCANn_base> + $023B_H$

**Value after reset:** $03FF\ FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CF17EMP | CF16EMP | CF15EMP | CF14EMP | CF13EMP | CF12EMP | CF11EMP | CF10EMP | CF9EMP | CF8EMP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF7EMP | CF6EMP | CF5EMP | CF4EMP | CF3EMP | CF2EMP | CF1EMP | CF0EMP | RF7EMP | RF6EMP | RF5EMP | RF4EMP | RF3EMP | RF2EMP | RF1EMP | RF0EMP |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.57 RSCANnFESTS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset is returned. |
| 25 | CF17EMP | Transmit/Receive FIFO Buffer Empty Status Flag<br>0: Transmit/receive FIFO buffer k contains a message.<br>1: Transmit/receive FIFO buffer k contains no message.<br>(k = 0 to 17) |
| 24 | CF16EMP | |
| 23 | CF15EMP | |
| 22 | CF14EMP | |
| 21 | CF13EMP | |
| 20 | CF12EMP | |
| 19 | CF11EMP | |
| 18 | CF10EMP | |
| 17 | CF9EMP | |
| 16 | CF8EMP | |
| 15 | CF7EMP | |
| 14 | CF6EMP | |
| 13 | CF5EMP | |
| 12 | CF4EMP | |
| 11 | CF3EMP | |
| 10 | CF2EMP | |
| 9 | CF1EMP | |
| 8 | CF0EMP | |

Table 21.57 RSCANnFESTS Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RF7EMP | Receive FIFO Buffer Empty Status Flag<br>0: Receive FIFO buffer x contains an unread message.<br>1: Receive FIFO buffer x contains no unread message (buffer empty).<br>(x = 0 to 7) |
| 6 | RF6EMP | |
| 5 | RF5EMP | |
| 4 | RF4EMP | |
| 3 | RF3EMP | |
| 2 | RF2EMP | |
| 1 | RF1EMP | |
| 0 | RF0EMP | |

The RSCANnFESTS register is set to 03FF $FFFF_H$ in global reset mode.

### CFkEMP Flag (k = 0 to 17)

The CFkEMP flag is set to 1 when the CFEMP flag in the RSCANnCFSTSk register is set to 1 (the transmit/receive FIFO buffer contains no message (buffer empty)). When the CFEMP flag is cleared to 0 (the transmit/receive FIFO buffer contains messages), the CFkEMP flag is cleared to 0.

### RFxEMP Flag (x = 0 to 7)

The RFxEMP flag is set to 1 when the RFEMP flag in the RSCANnRFSTSx register is set to 1 (the receive FIFO buffer contains no unread message). When the RFEMP flag is cleared to 0 (the receive FIFO buffer contains unread messages), the RFxEMP flag is cleared to 0.

### 21.4.8.2 RSCANnFFSTS — FIFO Full Status Register

**Access:** RSCANnFFSTS register is a read-only register that can be read in 32-bit units
RSCANnFFSTSL, RSCANnFFSTSH registers are read-only registers that can be read in 16-bit units
RSCANnFFSTSLL, RSCANnFFSTSLH, RSCANnFFSTSHL, RSCANnFFSTSHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnFFSTS: <RSCANn_base> + $023C_H$

RSCANnFFSTSL: <RSCANn_base> + $023C_H$,
RSCANnFFSTSH: <RSCANn_base> + $023E_H$

RSCANnFFSTSLL: <RSCANn_base> + $023C_H$,
RSCANnFFSTSLH: <RSCANn_base> + $023D_H$,
RSCANnFFSTSHL: <RSCANn_base> + $023E_H$,
RSCANnFFSTSHH: <RSCANn_base> + $023F_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CF17FLL | CF16FLL | CF15FLL | CF14FLL | CF13FLL | CF12FLL | CF11FLL | CF10FLL | CF9FLL | CF8FLL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF7FLL | CF6FLL | CF5FLL | CF4FLL | CF3FLL | CF2FLL | CF1FLL | CF0FLL | RF7FLL | RF6FLL | RF5FLL | RF4FLL | RF3FLL | RF2FLL | RF1FLL | RF0FLL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.58 RSCANnFFSTS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset is returned. |
| 25 | CF17FLL | Transmit/Receive FIFO Buffer Full Status Flag<br>0: Transmit/receive buffer k is not full.<br>1: Transmit/receive buffer k is full.<br>(k = 0 to 17) |
| 24 | CF16FLL | |
| 23 | CF15FLL | |
| 22 | CF14FLL | |
| 21 | CF13FLL | |
| 20 | CF12FLL | |
| 19 | CF11FLL | |
| 18 | CF10FLL | |
| 17 | CF9FLL | |
| 16 | CF8FLL | |
| 15 | CF7FLL | |
| 14 | CF6FLL | |
| 13 | CF5FLL | |
| 12 | CF4FLL | |
| 11 | CF3FLL | |
| 10 | CF2FLL | |
| 9 | CF1FLL | |
| 8 | CF0FLL | |

Table 21.58 RSCANnFFSTS Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RF7FLL | Receive FIFO Buffer Full Status Flag<br>0: Receive FIFO buffer x is not full.<br>1: Receive FIFO buffer x is full.<br>(x = 0 to 7) |
| 6 | RF6FLL | |
| 5 | RF5FLL | |
| 4 | RF4FLL | |
| 3 | RF3FLL | |
| 2 | RF2FLL | |
| 1 | RF1FLL | |
| 0 | RF0FLL | |

The RSCANnFFSTS register is cleared to 0000 $0000_H$ in global reset mode.

### CFkFLL Flag (k = 0 to 17)

The CFkFLL flag is set to 1 when the CFFLL flag in the RSCANnCFSTSk register is set to 1 (the transmit/receive FIFO buffer is full).
When the CFFLL flag is cleared to 0 (the transmit/receive FIFO buffer is not full), the CFkFLL flag is cleared to 0.

### RFxFLL Flag (x = 0 to 7)

The RFxFLL flag is set to 1 when the RFFLL flag in the RSCANnRFSTSx register is set to 1 (the receive FIFO buffer is full). When the RFFLL flag is cleared to 0 (the receive FIFO buffer is not full), the RFxFLL flag is cleared to 0.

### 21.4.8.3 RSCANnFMSTS — FIFO Message Lost Status Register

**Access:** RSCANnFMSTS register is a read-only register that can be read in 32-bit units
RSCANnFMSTSL, RSCANnFMSTSH registers are read-only registers that can be read in 16-bit units
RSCANnFMSTSLL, RSCANnFMSTSLH, RSCANnFMSTSHL, RSCANnFMSTSHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnFMSTS: <RSCANn_base> + $0240_H$

RSCANnFMSTSL: <RSCANn_base> + $0240_H$,
RSCANnFMSTSH: <RSCANn_base> + $0242_H$

RSCANnFMSTSLL: <RSCANn_base> + $0240_H$,
RSCANnFMSTSLH: <RSCANn_base> + $0241_H$,
RSCANnFMSTSHL: <RSCANn_base> + $0242_H$,
RSCANnFMSTSHH: <RSCANn_base> + $0243_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CF17MLT | CF16MLT | CF15MLT | CF14MLT | CF13MLT | CF12MLT | CF11MLT | CF10MLT | CF9MLT | CF8MLT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF7MLT | CF6MLT | CF5MLT | CF4MLT | CF3MLT | CF2MLT | CF1MLT | CF0MLT | RF7MLT | RF6MLT | RF5MLT | RF4MLT | RF3MLT | RF2MLT | RF1MLT | RF0MLT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.59 RSCANnFMSTS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset is returned. |
| 25 | CF17MLT | Transmit/Receive FIFO Buffer Message Lost Status Flag<br>0: No transmit/receive FIFO buffer k message is lost.<br>1: A transmit/receive FIFO buffer k message is lost.<br>(k = 0 to 17) |
| 24 | CF16MLT | |
| 23 | CF15MLT | |
| 22 | CF14MLT | |
| 21 | CF13MLT | |
| 20 | CF12MLT | |
| 19 | CF11MLT | |
| 18 | CF10MLT | |
| 17 | CF9MLT | |
| 16 | CF8MLT | |
| 15 | CF7MLT | |
| 14 | CF6MLT | |
| 13 | CF5MLT | |
| 12 | CF4MLT | |
| 11 | CF3MLT | |
| 10 | CF2MLT | |
| 9 | CF1MLT | |
| 8 | CF0MLT | |

Table 21.59 RSCANnFMSTS Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RF7MLT | Receive FIFO Buffer Message Lost Status Flag<br>0: No receive FIFO buffer x message is lost.<br>1: A receive FIFO buffer x message is lost.<br>(x = 0 to 7) |
| 6 | RF6MLT | |
| 5 | RF5MLT | |
| 4 | RF4MLT | |
| 3 | RF3MLT | |
| 2 | RF2MLT | |
| 1 | RF1MLT | |
| 0 | RF0MLT | |

The RSCANnFMSTS register is cleared to 0000 $0000_H$ in global reset mode.

### CFkMLT Flag (k = 0 to 17)

The CFkMLT flag is set to 1 when the CFMLT flag in the RSCANnCFSTSk register is set to 1 (a transmit/receive FIFO message is lost). When the CFMLT flag is cleared to 0, the CFkMLT flag is cleared to 0.

### RFxMLT Flag (x = 0 to 7)

The RFxMLT flag is set to 1 when the RFMLT flag in the RSCANnRFSTSx register is set to 1 (a receive FIFO message is lost). When the RFMLT flag is cleared to 0, the RFxMLT flag is cleared to 0.

### 21.4.8.4 RSCANnRFISTS — Receive FIFO Buffer Interrupt Flag Status Register

**Access:** RSCANnRFISTS register is a read-only register that can be read in 32-bit units
RSCANnRFISTSL register is a read-only register that can be read in 16-bit units
RSCANnRFISTSLL register is a read-only register that can be read in 8-bit units

**Address:** RSCANnRFISTS: <RSCANn_base> + $0244_H$

RSCANnRFISTSL: <RSCANn_base> + $0244_H$

RSCANnRFISTSLL: <RSCANn_base> + $0244_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RF7IF | RF6IF | RF5IF | RF4IF | RF3IF | RF2IF | RF1IF | RF0IF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.60 RSCANnRFISTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned. |
| 7 | RF7IF | Receive FIFO Buffer Interrupt Request Status Flag<br>0: No receive FIFO buffer x interrupt request is present.<br>1: A receive FIFO buffer x interrupt request is present.<br>(x = 0 to 7) |
| 6 | RF6IF | |
| 5 | RF5IF | |
| 4 | RF4IF | |
| 3 | RF3IF | |
| 2 | RF2IF | |
| 1 | RF1IF | |
| 0 | RF0IF | |

The RSCANnRFISTS register is cleared to $0000\ 0000_H$ in global reset mode.

#### RFxIF Flag (x = 0 to 7)

The RFxIF flag is set to 1 when the RFIF flag in the RSCANnRFSTSx register is set to 1 (a receive FIFO interrupt request is present). When the RFIF flag is cleared to 0, the RFxIF flag is cleared to 0.

### 21.4.8.5 RSCANnCFRISTS — Transmit/receive FIFO Buffer Receive Interrupt Flag Status Register

**Access:** RSCANnCFRISTS register is a read-only register that can be read in 32-bit units
RSCANnCFRISTSL, RSCANnCFRISTSH registers are read-only registers that can be read in 16-bit units
RSCANnCFRISTSLL, RSCANnCFRISTSLH, RSCANnCFRISTSHL registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnCFRISTS: <RSCANn_base> + $0248_H$

RSCANnCFRISTSL: <RSCANn_base> + $0248_H$,
RSCANnCFRISTSH: <RSCANn_base> + $024A_H$

RSCANnCFRISTSLL: <RSCANn_base> + $0248_H$,
RSCANnCFRISTSLH: <RSCANn_base> + $0249_H$,
RSCANnCFRISTSHL: <RSCANn_base> + $024A_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CF17RXIF | CF16RXIF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF15RXIF | CF14RXIF | CF13RXIF | CF12RXIF | CF11RXIF | CF10RXIF | CF9RXIF | CF8RXIF | CF7RXIF | CF6RXIF | CF5RXIF | CF4RXIF | CF3RXIF | CF2RXIF | CF1RXIF | CF0RXIF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.61 RSCANnCFRISTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 18 | Reserved | When read, the value after reset is returned. |
| 17 | CF17RXIF | Transmit/Receive FIFO Buffer Receive Interrupt Request Status Flag<br>0: No transmit/receive FIFO buffer k receive interrupt request is present.<br>1: A transmit/receive FIFO buffer k receive interrupt request is present.<br>(k = 0 to 17) |
| 16 | CF16RXIF | |
| 15 | CF15RXIF | |
| 14 | CF14RXIF | |
| 13 | CF13RXIF | |
| 12 | CF12RXIF | |
| 11 | CF11RXIF | |
| 10 | CF10RXIF | |
| 9 | CF9RXIF | |
| 8 | CF8RXIF | |
| 7 | CF7RXIF | |
| 6 | CF6RXIF | |
| 5 | CF5RXIF | |
| 4 | CF4RXIF | |
| 3 | CF3RXIF | |
| 2 | CF2RXIF | |
| 1 | CF1RXIF | |
| 0 | CF0RXIF | |

The RSCANnCFRISTS register is cleared to $0000\ 0000_H$ in global reset mode.

**CFkRXIF Flag (k = 0 to 17)**

The CFkRXIF flag is set to 1 when the CFRXIF flag in the RSCANnCFSTSk register is set to 1 (a transmit/receive FIFO receive interrupt request is present). When the CFRXIF flag is cleared to 0, the CFkRXIF flag is cleared to 0.

### 21.4.8.6 RSCANnCFTISTS — Transmit/receive FIFO Buffer Transmit Interrupt Flag Status Register

**Access:** RSCANnCFTISTS register is a read-only register that can be read in 32-bit units
RSCANnCFTISTSL, RSCANnCFTISTSH registers are read-only registers that can be read in 16-bit units
RSCANnCFTISTSLL, RSCANnCFTISTSLH, RSCANnCFTISTSHL registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnCFTISTS: <RSCANn_base> + $024C_H$

RSCANnCFTISTSL: <RSCANn_base> + $024C_H$,
RSCANnCFTISTSH: <RSCANn_base> + $024E_H$

RSCANnCFTISTSLL: <RSCANn_base> + $024C_H$,
RSCANnCFTISTSLH: <RSCANn_base> + $024D_H$,
RSCANnCFTISTSHL: <RSCANn_base> + $024E_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CF17TXIF | CF16TXIF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF15TXIF | CF14TXIF | CF13TXIF | CF12TXIF | CF11TXIF | CF10TXIF | CF9TXIF | CF8TXIF | CF7TXIF | CF6TXIF | CF5TXIF | CF4TXIF | CF3TXIF | CF2TXIF | CF1TXIF | CF0TXIF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.62 RSCANnCFTISTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 18 | Reserved | When read, the value after reset is returned. |
| 17 | CF17TXIF | Transmit/Receive FIFO Buffer Transmit Interrupt Request Status Flag<br>0: No transmit/receive FIFO buffer k transmit interrupt request is present.<br>1: A transmit/receive FIFO buffer k transmit interrupt request is present.<br>(k = 0 to 17) |
| 16 | CF16TXIF | |
| 15 | CF15TXIF | |
| 14 | CF14TXIF | |
| 13 | CF13TXIF | |
| 12 | CF12TXIF | |
| 11 | CF11TXIF | |
| 10 | CF10TXIF | |
| 9 | CF9TXIF | |
| 8 | CF8TXIF | |
| 7 | CF7TXIF | |
| 6 | CF6TXIF | |
| 5 | CF5TXIF | |
| 4 | CF4TXIF | |
| 3 | CF3TXIF | |
| 2 | CF2TXIF | |
| 1 | CF1TXIF | |
| 0 | CF0TXIF | |

The RSCANnCFTISTS register is cleared to $0000\ 0000_H$ in global reset mode.

**CFkTXIF Flag (k = 0 to 17)**

The CFkTXIF flag is set to 1 when the CFTXIF flag in the RSCANnCFSTSk register is set to 1 (a transmit/receive FIFO transmit interrupt request is present). When the CFTXIF flag is cleared to 0, the CFkTXIF flag is cleared to 0.

## 21.4.9 Details of Transmit Buffer-Related Registers

### 21.4.9.1 RSCANnTMCp — Transmit Buffer Control Register (p = 0 to 95)

**Access:** RSCANnTMCp register can be read or written in 8-bit units

**Address:** RSCANnTMCp: <RSCANn_base> + $0250_H$ + ($01_H$ × p)

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TMOM | TMTAR | TMTR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W*1 | R/W*1 |

Note 1. The only effective value for writing to this bit is 1, which sets the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.63 RSCANnTMCp Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 | TMOM | One-Shot Transmission Enable<br>0: One-shot transmission is disabled.<br>1: One-shot transmission is enabled. |
| 1 | TMTAR | Transmit Abort Request<br>0: Transmit abort is not requested.<br>1: Transmit abort is requested. |
| 0 | TMTR | Transmit Request<br>0: Transmission is not requested.<br>1: Transmission is requested. |

When the RSCANnTMCp register meets any of the following conditions, set it to $00_H$.

- The RSCANnTMCp register corresponds to the transmit buffer number selected by the CFTML[3:0] bits in the RSCANnCFCCk register (p = m × 16 + the value of CFTML[3:0] bits).
- The RSCANnTMCp register corresponds to the transmit buffer allocated to the transmit queue by the TXQDC[3:0] bits in the RSCANnTXQCCm (m = 0 to 5) register (p = (m × 16 + 15) to (m × 16 + 15 – the value of TXQDC[3:0] bits)).

Bits in the RSCANnTMCp register are all cleared to 0 in channel reset mode. Modify the RSCANnTMCp register in channel communication mode or channel halt mode.

#### TMOM Bit

Setting this bit to 1 enables one-shot transmission. When transmission fails, retransmission defined in the CAN protocol is not performed.

Modify the TMOM bit when the TMTRM flag in the RSCANnTMSTSp register is set to 0. Set the TMOM bit to 1 together with the TMTR bit.

### TMTAR Bit

Setting this bit to 1 generates a transmit abort request for the message stored in the transmit buffer. However, a message that is being transmitted or one that will be transmitted next cannot be aborted.

The TMTAR bit can be set to 1 when TMTR bit is 1.

The TMTAR bit is cleared to 0 when any of the following conditions is met, but cannot be cleared by the program writing 0 to the bit.

- Transmission has been completed.
- Transmit abort has been completed.
- An error or arbitration lost has been detected.

If this bit becomes 0 at the same time as the program writes 1 to this bit, this bit becomes 0.

### TMTR Bit

Setting this bit to 1 transmits the message stored in the transmit buffer.

The TMTR bit is cleared to 0 when any of the following conditions is met, but cannot be cleared by the program writing 0 to the bit.

- Transmission has been completed.
- Transmit abort has been completed after the TMTAR bit was set to 1.
- An error or arbitration-lost has been detected with the TMOM bit set to 1.

Set the TMTR bit to 1 when the value of TMTRF[1:0] in the RSCANnTMSTSp register is $00_B$.

### 21.4.9.2 RSCANnTMSTSp — Transmit Buffer Status Register (p = 0 to 95)

**Access:** RSCANnTMSTSp register can be read or written in 8-bit units

**Address:** RSCANnTMSTSp: <RSCANn_base> + $02D0_H$ + ($01_H$ × p)

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | TMTARM | TMTRM | TMTRF[1:0] | | TMTSTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R |

**Table 21.64 RSCANnTMSTSp Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 5 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 4 | TMTARM | Transmit Buffer Transmit Abort Request Status Flag<br>0: No transmit abort request is present.<br>1: A transmit abort request is present. |
| 3 | TMTRM | Transmit Buffer Transmit Request Status Flag<br>0: No transmit request is present.<br>1: A transmit request is present. |
| 2, 1 | TMTRF[1:0] | Transmit Buffer Transmit Result Status Flag<br>b2 b1<br>0 0: Transmission is in progress or no transmit request is present.<br>0 1: Transmit abort has been completed.<br>1 0: Transmission has been completed (without transmit abort request).<br>1 1: Transmission has been completed (with transmit abort request). |
| 0 | TMTSTS | Transmit Buffer Transmit Status Flag<br>0: Transmission is not in progress.<br>1: Transmission is in progress. |

The RSCANnTMSTSp register is cleared to all 0 in channel reset mode.

#### TMTARM Flag

The TMTARM flag is set to 1 when the TMTAR bit in the RSCANnTMCp register is set to 1.

The TMTARM flag is set to 0 when the TMTAR bit in the RSCANnTMCp register is set to 0.

#### TMTRM Flag

The TMTRM flag is set to 1 when the TMTR bit in the RSCANnTMCp register is set to 1.

The TMTRM flag is set to 0 when the TMTR bit in the RSCANnTMCp register is set to 0.

#### TMTRF[1:0] Flag

This flag indicates the result of transmission from the transmit buffer.

$00_B$: Transmission is in progress or no transmit request is present.

$01_B$: Transmission from the transmit buffer was aborted.

$10_B$: Transmission has been completed with the TMTAR bit in the RSCANnTMCp register set to 0 (transmit abort is not requested).

$11_B$: Transmission has been completed with the TMTAR bit in the RSCANnTMCp register set to 1 (transmit abort is requested).

Write $00_B$ to the TMTRF[1:0] flag in channel communication mode or channel halt mode. Do not write any value other than $00_B$ to this flag.

#### TMTSTS Flag

This flag is set to 1 when transmission from the transmit buffer starts, and is cleared to 0 when transmission from the transmit buffer has been completed or terminated due to a bus error or arbitration lost.

### 21.4.9.3 RSCANnTMIDp — Transmit Buffer ID Register (p = 0 to 95)

**Access:** RSCANnTMIDp register can be read or written in 32-bit units
RSCANnTMIDpL, RSCANnTMIDpH registers can be read or written in 16-bit units
RSCANnTMIDpLL, RSCANnTMIDpLH, RSCANnTMIDpHL, RSCANnTMIDpHH registers can be read or written in 8-bit units

**Address:** RSCANnTMIDp: <RSCANn_base> + $1000_H$ + ($10_H$ × p)

RSCANnTMIDpL: <RSCANn_base> + $1000_H$ + ($10_H$ × p),
RSCANnTMIDpH: <RSCANn_base> + $1002_H$ + ($10_H$ × p)

RSCANnTMIDpLL: <RSCANn_base> + $1000_H$ + ($10_H$ × p),
RSCANnTMIDpLH: <RSCANn_base> + $1001_H$ + ($10_H$ × p),
RSCANnTMIDpHL: <RSCANn_base> + $1002_H$ + ($10_H$ × p),
RSCANnTMIDpHH: <RSCANn_base> + $1003_H$ + ($10_H$ × p)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMIDE | TMRTR | THLEN | TMID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.65 RSCANnTMIDp Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | TMIDE | Transmit Buffer IDE<br>0: Standard ID<br>1: Extended ID |
| 30 | TMRTR | Transmit Buffer RTR<br>0: Data frame<br>1: Remote frame |
| 29 | THLEN | Transmit History Data Store Enable<br>0: Transmit history data is not stored in the buffer.<br>1: Transmit history data is stored in the buffer. |
| 28 to 0 | TMID[28:0] | Transmit Buffer ID Data<br>Set standard ID or extended ID.<br>For standard ID, write an ID to bits 10 to 0 and write 0 to bits 28 to 11. |

Modify this register when the TMTRM bit in the corresponding RSCANnTMSTSp register is set to 0 (no transmit request is present). If this register is linked to a transmit/receive FIFO buffer, do not write data to this register. If this register is allocated to the transmit queue, only write data to a transmit buffer p (p = m × 16 + 15) for the corresponding channel.

#### TMIDE Bit

This bit is used to set the ID format of the message to be transmitted from the transmit buffer.

#### TMRTR Bit

This bit is used to set the data format of the message to be transmitted from the transmit buffer.

#### THLEN Bit

When this bit is set to 1, the transmit history data (label information, buffer number, and buffer type, and timestamp (timestamp is included if the TMTSCE bit in the RSCANnGCFG register is 1)) of transmit messages is stored in the transmit history buffer after transmission is completed.

#### TMID[28:0] Bits

These bits are used to set the ID of the message to be transmitted from the transmit buffer.

### 21.4.9.4 RSCANnTMPTRp — Transmit Buffer Pointer Register (p = 0 to 95)

**Access:** RSCANnTMPTRp register can be read or written in 32-bit units
RSCANnTMPTRpH register can be read or written in 16-bit units
RSCANnTMPTRpHL, RSCANnTMPTRpHH registers can be read or written in 8-bit units

**Address:** RSCANnTMPTRp: <RSCANn_base> + $1004_H$ + ($10_H$ × p)

RSCANnTMPTRpH: <RSCANn_base> + $1006_H$ + ($10_H$ × p)

RSCANnTMPTRpHL: <RSCANn_base> + $1006_H$ + ($10_H$ × p),
RSCANnTMPTRpHH: <RSCANn_base> + $1007_H$ + ($10_H$ × p)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMDLC[3:0] | | | | — | — | — | — | TMPTR[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.66 RSCANnTMPTRp Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | TMDLC[3:0] | Transmit Buffer DLC Data<br>b31 b30 b29 b28<br>0 0 0 0: 0 data bytes<br>0 0 0 1: 1 data byte<br>0 0 1 0: 2 data bytes<br>0 0 1 1: 3 data bytes<br>0 1 0 0: 4 data bytes<br>0 1 0 1: 5 data bytes<br>0 1 1 0: 6 data bytes<br>0 1 1 1: 7 data bytes<br>1 X X X: 8 data bytes |
| 27 to 24 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 23 to 16 | TMPTR[7:0] | Transmit Buffer Label Data<br>Set the label information to be stored in the transmit history buffer. |
| 15 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

Modify this register when the TMTRM bit in the corresponding RSCANnTMSTSp register is set to 0 (no transmit request is present). If this register is linked to a transmit/receive FIFO buffer, do not write to this register. If this register is allocated to the transmit queue, only write to a transmit buffer p (p = m × 16 + 15) for the corresponding channel.

#### TMDLC[3:0] Bits

These bits are used to set the data length of the message to be transmitted from the transmit buffer when the TMRTR bit in the RSCANnTMIDp register is set to 0 (data frame). If the data length is set to $1001_B$ or more, the transmit data is 8 bytes long.

When the TMRTR bit is set to 1 (remote frame), set the data length of messages to be requested.

**TMPTR[7:0] Bits**

When message transmission has been completed, the TMPTR[7:0] value is stored in the transmit history buffer.

### 21.4.9.5 RSCANnTMDF0p — Transmit Buffer Data Field 0 Register (p = 0 to 95)

**Access:** RSCANnTMDF0p register can be read or written in 32-bit units
RSCANnTMDF0pL, RSCANnTMDF0pH registers can be read or written in 16-bit units
RSCANnTMDF0pLL, RSCANnTMDF0pLH, RSCANnTMDF0pHL, RSCANnTMDF0pHH registers can be read or written in 8-bit units

**Address:** RSCANnTMDF0p: <RSCANn_base> + $1008_H$ + ($10_H$ × p)

RSCANnTMDF0pL: <RSCANn_base> + $1008_H$ + ($10_H$ × p),
RSCANnTMDF0pH: <RSCANn_base> + $100A_H$ + ($10_H$ × p)

RSCANnTMDF0pLL: <RSCANn_base> + $1008_H$ + ($10_H$ × p),
RSCANnTMDF0pLH: <RSCANn_base> + $1009_H$ + ($10_H$ × p),
RSCANnTMDF0pHL: <RSCANn_base> + $100A_H$ + ($10_H$ × p),
RSCANnTMDF0pHH: <RSCANn_base> + $100B_H$ + ($10_H$ × p)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMDB3[7:0] | | | | | | | | TMDB2[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMDB1[7:0] | | | | | | | | TMDB0[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.67 RSCANnTMDF0p Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | TMDB3[7:0] | Transmit Buffer Data Byte 3 |
| 23 to 16 | TMDB2[7:0] | Transmit Buffer Data Byte 2 |
| 15 to 8 | TMDB1[7:0] | Transmit Buffer Data Byte 1 |
| 7 to 0 | TMDB0[7:0] | Transmit Buffer Data Byte 0<br><br>Set the transmit buffer data. |

Modify this register when the TMTRM bit in the corresponding RSCANnTMSTSp register is set to 0 (no transmit request is present). If this register is linked to a transmit/receive FIFO buffer, do not write to this register. If this register is allocated to the transmit queue, only write to a transmit buffer p (p = m × 16 + 15) for the corresponding channel.

### 21.4.9.6 RSCANnTMDF1p — Transmit Buffer Data Field 1 Register (p = 0 to 95)

**Access:** RSCANnTMDF1p register can be read or written in 32-bit units
RSCANnTMDF1pL, RSCANnTMDF1pH registers can be read or written in 16-bit units
RSCANnTMDF1pLL, RSCANnTMDF1pLH, RSCANnTMDF1pHL, RSCANnTMDF1pHH registers can be read or written in 8-bit units

**Address:** RSCANnTMDF1p: <RSCANn_base> + $100C_H$ + ($10_H$ × p)

RSCANnTMDF1pL: <RSCANn_base> + $100C_H$ + ($10_H$ × p),
RSCANnTMDF1pH: <RSCANn_base> + $100E_H$ + ($10_H$ × p)

RSCANnTMDF1pLL: <RSCANn_base> + $100C_H$ + ($10_H$ × p),
RSCANnTMDF1pLH: <RSCANn_base> + $100D_H$ + ($10_H$ × p),
RSCANnTMDF1pHL: <RSCANn_base> + $100E_H$ + ($10_H$ × p),
RSCANnTMDF1pHH: <RSCANn_base> + $100F_H$ + ($10_H$ × p)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMDB7[7:0] | | | | | | | | TMDB6[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMDB5[7:0] | | | | | | | | TMDB4[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.68 RSCANnTMDF1p Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | TMDB7[7:0] | Transmit Buffer Data Byte 7 |
| 23 to 16 | TMDB6[7:0] | Transmit Buffer Data Byte 6 |
| 15 to 8 | TMDB5[7:0] | Transmit Buffer Data Byte 5 |
| 7 to 0 | TMDB4[7:0] | Transmit Buffer Data Byte 4<br><br>Set the transmit buffer data. |

Modify this register when the TMTRM bit in the corresponding RSCANnTMSTSp register is set to 0 (no transmit request is present). If this register is linked to a transmit/receive FIFO buffer, do not write to this register. If this register is allocated to the transmit queue, only write to a transmit buffer p (p = m × 16 + 15) for the corresponding channel.

### 21.4.9.7 RSCANnTMIECy — Transmit Buffer Interrupt Enable Configuration Register (y = 0 to 2)

**Access:** RSCANnTMIECy register can be read or written in 32-bit units
RSCANnTMIECyL, RSCANnTMIECyH registers can be read or written in 16-bit units
RSCANnTMIECyLL, RSCANnTMIECyLH, RSCANnTMIECyHL, RSCANnTMIECyHH registers can be read or written in 8-bit units

**Address:** RSCANnTMIECy: <RSCANn_base> + $0390_H$ + ($04_H$ × y)

RSCANnTMIECyL: <RSCANn_base> + $0390_H$ + ($04_H$ × y),
RSCANnTMIECyH: <RSCANn_base> + $0392_H$ + ($04_H$ × y)

RSCANnTMIECyLL: <RSCANn_base> + $0390_H$ + ($04_H$ × y),
RSCANnTMIECyLH: <RSCANn_base> + $0391_H$ + ($04_H$ × y),
RSCANnTMIECyHL: <RSCANn_base> + $0392_H$ + ($04_H$ × y),
RSCANnTMIECyHH: <RSCANn_base> + $0393_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMIEp (p = y × 32 + 31 to y × 32 + 16 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMIEp (p = y × 32 + 15 to y × 32 + 0 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.69 RSCANnTMIECy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | TMIEp | Transmit Buffer Interrupt Enable p (p = y × 32 + 31 to y × 32 + 16)<br>0: Transmit buffer interrupt is disabled<br>1: Transmit buffer interrupt is enabled |
| 15 to 0 | TMIEp | Transmit Buffer Interrupt Enable p (p = y × 32 + 15 to y × 32 + 0)<br>0: Transmit buffer interrupt is disabled.<br>1: Transmit buffer interrupt is enabled. |

#### TMIEp Bits (p = 0 to 95)

When any of these bits is set to 1 and the corresponding transmission has been completed, a transmit buffer interrupt request is generated.

Modify these bits when the TMTRM flag in the corresponding RSCANnTMSTSp register is 0 (no transmit request is present).

Write 0 to bits corresponding to transmit buffers linked to transmit/receive FIFO buffers or transmit buffers allocated to the transmit queue.

**Table 21.70** shows the bit assignment.

**Table 21.70 TMIEp Bit Assignment**

| Bit Position | Channel | Transmit Buffer Number |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| . . | . . | . . |
| 15 | 0 | 15 |
| 16 | 1 | 0 |
| . . | . . | . . |
| 30 | 1 | 14 |
| 31 | 1 | 15 |
| 32 | 2 | 0 |
| 33 | 2 | 1 |
| . . | . . | . . |
| 47 | 2 | 15 |
| 48 | 3 | 0 |
| . . | . . | . . |
| 62 | 3 | 14 |
| 63 | 3 | 15 |
| 64 | 4 | 0 |
| 65 | 4 | 1 |
| . . | . . | . . |
| 78 | 4 | 14 |
| 79 | 4 | 15 |
| 80 | 5 | 0 |
| 81 | 5 | 1 |
| . . | . . | . . |
| 94 | 5 | 14 |
| 95 | 5 | 15 |

## 21.4.10 Details of Transmit Buffer Status-Related Registers

### 21.4.10.1 RSCANnTMTRSTSy — Transmit Buffer Transmit Request Status Register (y = 0 to 2)

**Access:** RSCANnTMTRSTSy register is a read-only register that can be read in 32-bit units
RSCANnTMTRSTSyL, RSCANnTMTRSTSyH registers are read-only registers that can be read in 16-bit units
RSCANnTMTRSTSyLL, RSCANnTMTRSTSyLH, RSCANnTMTRSTSyHL, RSCANnTMTRSTSyHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnTMTRSTSy: <RSCANn_base> + $0350_H$ + ($04_H$ × y)

RSCANnTMTRSTSyL: <RSCANn_base> + $0350_H$ + ($04_H$ × y),
RSCANnTMTRSTSyH: <RSCANn_base> + $0352_H$ + ($04_H$ × y)

RSCANnTMTRSTSyLL: <RSCANn_base> + $0350_H$ + ($04_H$ × y),
RSCANnTMTRSTSyLH: <RSCANn_base> + $0351_H$ + ($04_H$ × y),
RSCANnTMTRSTSyHL: <RSCANn_base> + $0352_H$ + ($04_H$ × y),
RSCANnTMTRSTSyHH: <RSCANn_base> + $0353_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTRSTSp (p = y × 32 + 31 to y × 32 + 16 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTRSTSp (p = y × 32 + 15 to y × 32 + 0 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.71 RSCANnTMTRSTSy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | TMTRSTSp | Transmit Buffer Transmit Request Status Flag p (p = y × 32 + 31 to y × 32 + 16)<br>0: No transmit request is present.<br>1: A transmit request is present. |
| 15 to 0 | TMTRSTSp | Transmit Buffer Transmit Request Status Flag p (p = y × 32 + 15 to y × 32 + 0)<br>0: No transmit request is present.<br>1: A transmit request is present. |

#### TMTRSTSp Flags (p = 0 to 95)

These flags indicate the status of the TMTR bit in the RSCANnTMCp register.

When the TMTR bit is set to 1 (transmission is requested), the corresponding TMTRSTSp flag is set to 1.

The corresponding TMTRSTSp flag is cleared to 0 when the TMTR bit is set to 0 (transmission is not requested) or in channel reset mode.

**Table 21.72** shows the bit assignment.

**Table 21.72 TMTRSTSp Bit Assignment**

| Bit Position | Channel | Transmit Buffer Number |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| . . | . . | . . |
| 15 | 0 | 15 |
| 16 | 1 | 0 |
| . . | . . | . . |
| 30 | 1 | 14 |
| 31 | 1 | 15 |
| 32 | 2 | 0 |
| 33 | 2 | 1 |
| . . | . . | . . |
| 47 | 2 | 15 |
| 48 | 3 | 0 |
| . . | . . | . . |
| 62 | 3 | 14 |
| 63 | 3 | 15 |
| 64 | 4 | 0 |
| 65 | 4 | 1 |
| . . | . . | . . |
| 78 | 4 | 14 |
| 79 | 4 | 15 |
| 80 | 5 | 0 |
| 81 | 5 | 1 |
| . . | . . | . . |
| 94 | 5 | 14 |
| 95 | 5 | 15 |

### 21.4.10.2 RSCANnTMTARSTSy — Transmit Buffer Transmit Abort Request Status Register (y = 0 to 2)

**Access:** RSCANnTMTARSTSy register is a read-only register that can be read in 32-bit units
RSCANnTMTARSTSyL, RSCANnTMTARSTSyH registers are read-only registers that can be read in 16-bit units
RSCANnTMTARSTSyLL, RSCANnTMTARSTSyLH, RSCANnTMTARSTSyHL, RSCANnTMTARSTSyHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnTMTARSTSy: <RSCANn_base> + $0360_H$ + ($04_H$ × y)

RSCANnTMTARSTSyL: <RSCANn_base> + $0360_H$ + ($04_H$ × y),
RSCANnTMTARSTSyH: <RSCANn_base> + $0362_H$ + ($04_H$ × y)

RSCANnTMTARSTSyLL: <RSCANn_base> + $0360_H$ + ($04_H$ × y),
RSCANnTMTARSTSyLH: <RSCANn_base> + $0361_H$ + ($04_H$ × y),
RSCANnTMTARSTSyHL: <RSCANn_base> + $0362_H$ + ($04_H$ × y),
RSCANnTMTARSTSyHH: <RSCANn_base> + $0363_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTARSTSp (p = y × 32 + 31 to y × 32 + 16 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTARSTSp (p = y × 32 + 15 to y × 32 + 0 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.73 RSCANnTMTARSTSy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | TMTARSTSp | Transmit Buffer Transmit Abort Request Status Flag p (p = y × 32 + 31 to y × 32 + 16)<br>0: No transmit abort request is present.<br>1: A transmit abort request is present. |
| 15 to 0 | TMTARSTSp | Transmit Buffer Transmit Abort Request Status Flag p (p = y × 32 + 15 to y × 32 + 0)<br>0: No transmit abort request is present.<br>1: A transmit abort request is present. |

#### TMTARSTSp Flags (p = 0 to 95)

These flags indicate the status of the TMTAR bit in the RSCANnTMCp register.

When the TMTAR bit is set to 1 (transmit abort is requested), the corresponding TMTARSTSp flag is set to 1.

The corresponding TMTARSTSp flag is cleared to 0 when the TMTAR bit is set to 0 (transmit abort is not requested) or in channel reset mode.

**Table 21.74** shows the bit assignment.

Table 21.74 TMTARSTSp Bit Assignment

| Bit Position | Channel | Transmit Buffer Number |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| . . | . . | . . |
| 15 | 0 | 15 |
| 16 | 1 | 0 |
| . . | . . | . . |
| 30 | 1 | 14 |
| 31 | 1 | 15 |
| 32 | 2 | 0 |
| 33 | 2 | 1 |
| . . | . . | . . |
| 47 | 2 | 15 |
| 48 | 3 | 0 |
| . . | . . | . . |
| 62 | 3 | 14 |
| 63 | 3 | 15 |
| 64 | 4 | 0 |
| 65 | 4 | 1 |
| . . | . . | . . |
| 78 | 4 | 14 |
| 79 | 4 | 15 |
| 80 | 5 | 0 |
| 81 | 5 | 1 |
| . . | . . | . . |
| 94 | 5 | 14 |
| 95 | 5 | 15 |

### 21.4.10.3 RSCANnTMTCSTSy — Transmit Buffer Transmit Complete Status Register (y = 0 to 2)

**Access:** RSCANnTMTCSTSy register is a read-only register that can be read in 32-bit units
RSCANnTMTCSTSyL, RSCANnTMTCSTSyH registers are read-only registers that can be read in 16-bit units
RSCANnTMTCSTSyLL, RSCANnTMTCSTSyLH, RSCANnTMTCSTSyHL, RSCANnTMTCSTSyHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnTMTCSTSy: <RSCANn_base> + $0370_H$ + ($04_H$ × y)

RSCANnTMTCSTSyL: <RSCANn_base> + $0370_H$ + ($04_H$ × y),
RSCANnTMTCSTSyH: <RSCANn_base> + $0372_H$ + ($04_H$ × y)

RSCANnTMTCSTSyLL: <RSCANn_base> + $0370_H$ + ($04_H$ × y),
RSCANnTMTCSTSyLH: <RSCANn_base> + $0371_H$ + ($04_H$ × y),
RSCANnTMTCSTSyHL: <RSCANn_base> + $0372_H$ + ($04_H$ × y),
RSCANnTMTCSTSyHH: <RSCANn_base> + $0373_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTCSTSp (p = y × 32 + 31 to y × 32 + 16 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTCSTSp (p = y × 32 + 15 to y × 32 + 0 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.75 RSCANnTMTCSTSy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | TMTCSTSp | Transmit Buffer Transmit Complete Status Flag p (p = y × 32 + 31 to y × 32 + 16)<br>0: Transmission has not been completed.<br>1: Transmission has been completed. |
| 15 to 0 | TMTCSTSp | Transmit Buffer Transmit Complete Status Flag p (p = y × 32 + 15 to y × 32 + 0)<br>0: Transmission has not been completed.<br>1: Transmission has been completed. |

#### TMTCSTSp Flags (p = 0 to 95)

When the TMTRF[1:0] flag in the RSCANnTMSTSp register is set to $10_B$ (transmission has been completed (without transmit abort request)) or $11_B$ (transmission has been completed (with transmit abort request)), the corresponding TMTCSTSp flag is set to 1.

A TMTCSTSp flag is cleared to 0 when the corresponding TMTRF[1:0] flag is set to $00_B$ or in channel reset mode.

**Table 21.76** shows the bit assignment.

Table 21.76 TMTCSTSp Bit Assignment

| Bit Position | Channel | Transmit Buffer Number |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| . . | . . | . . |
| 15 | 0 | 15 |
| 16 | 1 | 0 |
| . . | . . | . . |
| 30 | 1 | 14 |
| 31 | 1 | 15 |
| 32 | 2 | 0 |
| 33 | 2 | 1 |
| . . | . . | . . |
| 47 | 2 | 15 |
| 48 | 3 | 0 |
| . . | . . | . . |
| 62 | 3 | 14 |
| 63 | 3 | 15 |
| 64 | 4 | 0 |
| 65 | 4 | 1 |
| . . | . . | . . |
| 78 | 4 | 14 |
| 79 | 4 | 15 |
| 80 | 5 | 0 |
| 81 | 5 | 1 |
| . . | . . | . . |
| 94 | 5 | 14 |
| 95 | 5 | 15 |

### 21.4.10.4 RSCANnTMTASTSy — Transmit Buffer Transmit Abort Status Register (y = 0 to 2)

**Access:** RSCANnTMTASTSy register is a read-only register that can be read in 32-bit units
RSCANnTMTASTSyL, RSCANnTMTASTSyH registers are read-only registers that can be read in 16-bit units
RSCANnTMTASTSyLL, RSCANnTMTASTSyLH, RSCANnTMTASTSyHL, RSCANnTMTASTSyHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnTMTASTSy: <RSCANn_base> + $0380_H$ + ($04_H$ × y)

RSCANnTMTASTSyL: <RSCANn_base> + $0380_H$ + ($04_H$ × y),
RSCANnTMTASTSyH: <RSCANn_base> + $0382_H$ + ($04_H$ × y)

RSCANnTMTASTSyLL: <RSCANn_base> + $0380_H$ + ($04_H$ × y),
RSCANnTMTASTSyLH: <RSCANn_base> + $0381_H$ + ($04_H$ × y),
RSCANnTMTASTSyHL: <RSCANn_base> + $0382_H$ + ($04_H$ × y),
RSCANnTMTASTSyHH: <RSCANn_base> + $0383_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTASTSp (p = y × 32 + 31 to y × 32 + 16 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTASTSp (p = y × 32 + 15 to y × 32 + 0 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.77 RSCANnTMTASTSy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | TMTASTSp | Transmit Buffer Transmit Abort Status Flag p (p = y × 32 + 31 to y × 32 + 16)<br>0: Transmission is not aborted<br>1: Transmission is aborted |
| 15 to 0 | TMTASTSp | Transmit Buffer Transmit Abort Status Flag p (p = y × 32 + 15 to y × 32 + 0)<br>0: Transmission is not aborted.<br>1: Transmission is aborted. |

#### TMTASTSp Flags (p = 0 to 95)

When the TMTRF[1:0] flag in the RSCANnTMSTSp register is set to $01_B$ (transmit abort has been completed), the corresponding TMTASTSp flag is set to 1.

A TMTASTSp flag is cleared to 0 when the corresponding TMTRF[1:0] flag is set to $00_B$ or in channel reset mode.

**Table 21.78** shows the bit assignment.

Table 21.78 TMTASTSp Bit Assignment

| Bit Position | Channel | Transmit Buffer Number |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| . . | . . | . . |
| 15 | 0 | 15 |
| 16 | 1 | 0 |
| . . | . . | . . |
| 30 | 1 | 14 |
| 31 | 1 | 15 |
| 32 | 2 | 0 |
| 33 | 2 | 1 |
| . . | . . | . . |
| 47 | 2 | 15 |
| 48 | 3 | 0 |
| . . | . . | . . |
| 62 | 3 | 14 |
| 63 | 3 | 15 |
| 64 | 4 | 0 |
| 65 | 4 | 1 |
| . . | . . | . . |
| 78 | 4 | 14 |
| 79 | 4 | 15 |
| 80 | 5 | 0 |
| 81 | 5 | 1 |
| . . | . . | . . |
| 94 | 5 | 14 |
| 95 | 5 | 15 |

## 21.4.11 Details of Transmit Queue-Related Registers

### 21.4.11.1 RSCANnTXQCCm — Transmit Queue Configuration and Control Register (m = 0 to 5)

**Access:** RSCANnTXQCCm register can be read or written in 32-bit units
RSCANnTXQCCmL register can be read or written in 16-bit units
RSCANnTXQCCmLL, RSCANnTXQCCmLH registers can be read or written in 8-bit units

**Address:** RSCANnTXQCCm: <RSCANn_base> + $03A0_H$ + ($04_H$ × m)

RSCANnTXQCCmL: <RSCANn_base> + $03A0_H$ + ($04_H$ × m)

RSCANnTXQCCmLL: <RSCANn_base> + $03A0_H$ + ($04_H$ × m),
RSCANnTXQCCmLH: <RSCANn_base> + $03A1_H$ + ($04_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | TXQIM | TXQIE | TXQDC[3:0] | | | | — | — | — | — | — | — | — | TXQE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W |

**Table 21.79 RSCANnTXQCCm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 14 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 13 | TXQIM | Transmit Queue Interrupt Source Select<br>0: When the transmit queue becomes empty upon completion of message transmission, a transmit queue interrupt request is generated.<br>1: A transmit queue interrupt request is generated each time a message has been transmitted. |
| 12 | TXQIE | Transmit Queue Interrupt Enable<br>0: Transmit queue interrupt is disabled.<br>1: Transmit queue interrupt is enabled. |
| 11 to 8 | TXQDC[3:0] | Transmit Queue Depth Configuration<br>Setting these bits to g (g = 2 to 15) makes the (g + 1)-buffer transmit queue available.<br>Setting these bits to 0 disables the transmit queue.<br>Setting these bits to 1 is prohibited. |
| 7 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | TXQE | Transmit Queue Enable<br>0: The transmit queue is not used.<br>1: The transmit queue is used. |

#### TXQIM Bit

This bit is used to select a transmit queue interrupt source. Modify this bit in channel reset mode.

#### TXQIE Bit

When the TXQIE bit is set to 1 and the source selected by the TXQIM bit occurs, an interrupt request is generated.

Set the TXQE bit to 0 before modifying the TXQIE bit.

#### TXQDC[3:0] Bits

These bits are used to specify the number of transmit buffers to be allocated to the transmit queues. Transmit buffers are allocated to transmit queues in descending order of buffer number, that is, from (m × 16 + 15) to (m × 16 + 0). For examples of how buffer allocation is done, see **Figure 21.9**. Modify these bits only in channel reset mode.

#### TXQE Bit

Setting this bit to 1 makes the transmit queue available. Modify this bit in channel communication mode or channel halt mode. This bit is cleared to 0 in channel reset mode.

Before setting the TXQE bit to 1, set the TXQDC[3:0] bits to $0010_B$ or more.

### 21.4.11.2 RSCANnTXQSTSm — Transmit Queue Status Register (m = 0 to 5)

**Access:** RSCANnTXQSTSm register can be read or written in 32-bit units
RSCANnTXQSTSmL register can be read or written in 16-bit units
RSCANnTXQSTSmLL register can be read or written in 8-bit units

**Address:** RSCANnTXQSTSm: <RSCANn_base> + $03C0_H$ + ($04_H$ × m)

RSCANnTXQSTSmL: <RSCANn_base> + $03C0_H$ + ($04_H$ × m)

RSCANnTXQSTSmLL: <RSCANn_base> + $03C0_H$ + ($04_H$ × m)

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TXQIF | TXQFLL | TXQEMP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W*1 | R | R |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.80 RSCANnTXQSTSm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 to 8 | Reserved | When read, the undefined value is returned. When writing to these bits, write 0. |
| 7 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 | TXQIF | Transmit Queue Interrupt Request Flag<br>0: No transmit queue interrupt request is present.<br>1: A transmit queue interrupt request is present. |
| 1 | TXQFLL | Transmit Queue Full Status Flag<br>0: The transmit queue is not full.<br>1: The transmit queue is full. |
| 0 | TXQEMP | Transmit Queue Empty Status Flag<br>0: The transmit queue contains messages.<br>1: The transmit queue contains no message (transmit queue empty). |

#### TXQIF Flag

The TXQIF flag is set to 1 when the event specified by the TXQIM bit in the RSCANnTXQCCm register has occurred.

The TXQIF flag is cleared to 0 in channel reset mode or by writing 0 to this flag. This flag is not cleared to 0 by setting the TXQE bit in the RSCANnTXQCCm register to 0 (the transmit queue is not used).

#### TXQFLL Flag

The TXQFLL flag is set to 1 when the number of messages set for the transmit queue matches the transmit queue depth set by the TXQDC[3:0] bits in the RSCANnTXQCCm register.

This flag is cleared to 0 in any of the following cases.

- The number of messages set for the transmit queue is smaller than the transmit queue depth set by the TXQDC[3:0] bits.
- In channel reset mode

#### TXQEMP Flag

The TXQEMP flag is cleared to 0 when even a single message is set for the transmit queue.

This flag is set to 1 in any of the following cases.

- The TXQE bit is set to 0 (the transmit queue is not used).
- The transmit queue becomes empty.
- In channel reset mode

### 21.4.11.3 RSCANnTXQPCTRm — Transmit Queue Pointer Control Register (m = 0 to 5)

**Access:** RSCANnTXQPCTRm register is a write-only register that can be written in 32-bit units
RSCANnTXQPCTRmL register is a write-only register that can be written in 16-bit units
RSCANnTXQPCTRmLL register is a write-only register that can be written in 8-bit units

**Address:** RSCANnTXQPCTRm: <RSCANn_base> + $03E0_H$ + ($04_H$ × m)

RSCANnTXQPCTRmL: <RSCANn_base> + $03E0_H$ + ($04_H$ × m)

RSCANnTXQPCTRmLL: <RSCANn_base> + $03E0_H$ + ($04_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TXQPC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**Table 21.81 RSCANnTXQPCTRm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing, write the value after reset. |
| 7 to 0 | TXQPC[7:0] | Transmit Queue Pointer Control<br>Writing $FF_H$ to these bits moves the write pointer of the transmit queue to the next queue buffer. |

#### TXQPC[7:0] Bits

Writing $FF_H$ to the TXQPC[7:0] bits moves the write pointer to the next transmit queue buffer and generates a transmit request of the message. Write transmit messages to the RSCANnTMIDp, RSCANnTMPTRp, RSCANnTMDF0p, and RSCANnTMDF1p registers (p = 15, 31, 47, 63, 79, and 95) before writing $FF_H$ to the TXQPC[7:0] bits.

When writing $FF_H$ to these bits, make sure that the TXQE bit in the RSCANnTXQCCm register is set to 1 (the transmit queue is used) and the TXQFLL flag in the RSCANnTXQSTSm register is 0 (the transmit queue is not full).

## 21.4.12 Details of Transmit history-related Registers

### 21.4.12.1 RSCANnTHLCCm — Transmit History Configuration and Control Register (m = 0 to 5)

**Access:** RSCANnTHLCCm register can be read or written in 32-bit units
RSCANnTHLCCmL register can be read or written in 16-bit units
RSCANnTHLCCmLL, RSCANnTHLCCmLH registers can be read or written in 8-bit units

**Address:** RSCANnTHLCCm: <RSCANn_base> + $0400_H$ + ($04_H$ × m)

RSCANnTHLCCmL: <RSCANn_base> + $0400_H$ + ($04_H$ × m)

RSCANnTHLCCmLL: <RSCANn_base> + $0400_H$ + ($04_H$ × m),
RSCANnTHLCCmLH: <RSCANn_base> + $0401_H$ + ($04_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | THL DTE | THLIM | THLIE | — | — | — | — | — | — | — | THLE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W |

**Table 21.82 RSCANnTHLCCm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 10 | THLDTE | Transmit History Target Buffer Select<br>0: Entry from transmit/receive FIFO buffers and transmit queue<br>1: Entry from transmit buffers, transmit/receive FIFO buffers, and transmit queue |
| 9 | THLIM | Transmit History Interrupt Source Select<br>0: When 12 sets of data have been stored in the transmit history buffer<br>1: When a single set of transmit history data has been stored |
| 8 | THLIE | Transmit History Interrupt Enable<br>0: Transmit history interrupt is disabled.<br>1: Transmit history interrupt is enabled. |
| 7 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | THLE | Transmit History Buffer Enable<br>0: Transmit history buffer is not used.<br>1: Transmit history buffer is used. |

#### THLDTE Bit

When this bit is set to 0, the transmit history data of messages transmitted from transmit/receive FIFO buffers and the transmit queue is stored in the transmit history buffer. When this bit is set to 1, the transmit history data of messages transmitted from transmit buffers, transmit/receive FIFO buffers, and the transmit queue is stored in the transmit history buffer.

Modify this bit only in channel reset mode.

### THLIM Bit

This bit is used to select a transmit history interrupt source.

Modify this bit only in channel reset mode.

### THLIE Bit

When the THLIE bit is set to 1 and the source selected by the THLIM bit has occurred, a transmit history interrupt request is generated. Modify the THLIE bit only when the THLE bit is set to 0.

### THLE Bit

Setting this bit to 1 makes the transmit history buffer available. When data transmission from the buffer selected by the THLDTE bit has been completed, the transmit history data of transmit messages is stored in the transmit history buffer.

Modify this bit in channel communication mode or channel halt mode.

This bit is cleared to 0 in channel reset mode.

### 21.4.12.2 RSCANnTHLSTSm — Transmit History Status Register (m = 0 to 5)

**Access:** RSCANnTHLSTSm register can be read or written in 32-bit units
RSCANnTHLSTSmL register can be read or written in 16-bit units
RSCANnTHLSTSmLL register can be read or written in 8-bit units
RSCANnTHLSTSmLH register is a read-only register that can be read in 8-bit units

**Address:** RSCANnTHLSTSm: <RSCANn_base> + $0420_H$ + ($04_H$ × m)

RSCANnTHLSTSmL: <RSCANn_base> + $0420_H$ + ($04_H$ × m)

RSCANnTHLSTSmLL: <RSCANn_base> + $0420_H$ + ($04_H$ × m),
RSCANnTHLSTSmLH: <RSCANn_base> + $0421_H$ + ($04_H$ × m)

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | THLMC[4:0] | | | | | — | — | — | — | THLIF | THLELT | THLFLL | THL EMP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W*1 | R/W*1 | R | R |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.83 RSCANnTHLSTSm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 to 8 | THLMC[4:0] | Transmit History Buffer Unread Data Counter<br>These bits indicate the number of unread data sets stored in the transmit history buffer. |
| 7 to 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 | THLIF | Transmit History Interrupt Request Flag<br>0: No transmit history interrupt request is present.<br>1: A transmit history interrupt request is present. |
| 2 | THLELT | Transmit History Buffer Overflow Flag<br>0: Transmit history buffer overflow has not occurred.<br>1: Transmit history buffer overflow has occurred. |
| 1 | THLFLL | Transmit history Buffer Full Status Flag<br>0: Transmit history buffer is not full.<br>1: Transmit history buffer is full. |
| 0 | THLEMP | Transmit History Buffer Empty Status Flag<br>0: Transmit history buffer contains unread data.<br>1: Transmit history buffer contains no unread data (buffer empty). |

#### THLMC[4:0] Bits

These bits indicate the number of unread data sets stored in the transmit history buffer.

These bits are cleared to 0 in channel reset mode.

#### THLIF Flag

The THLIF flag is set to 1 when the interrupt source specified with the THLIM bit in the RSCANnTHLCCm register occurs.

This flag is cleared to 0 in channel reset mode or by the program writing 0 to this flag.

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared.

When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

#### THLELT Flag

The THLELT flag is set to 1 when an attempt is made to store new transmit history data while the transmit history buffer is full. In this case, the new data is discarded. This flag becomes 0 in channel reset mode or by the program writing 0 to this flag.

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared.

When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

#### THLFLL Flag

The THLFLL flag is set to 1 when 16 data sets have been stored in the transmit history buffer, and is cleared to 0 when the number of data sets stored in the transmit history buffer has decreased to less than 16. This bit is also cleared to 0 in channel reset mode or when the THLE bit in the RSCANnTHLCCm register is set to 0 (transmit history buffer is not used).

#### THLEMP Flag

The THLEMP flag is cleared to 0 when even a single set of transmit history data has been stored in the transmit history buffer.

This flag is set to 1 when all the data in the transmit history buffer has been read. This flag is also set to 1 in channel reset mode or when the THLE bit in the RSCANnTHLCCm register is set to 0 (transmit history buffer is not used).

### 21.4.12.3 RSCANnTHLPCTRm — Transmit History Pointer Control Register (m = 0 to 5)

**Access:** RSCANnTHLPCTRm register is a write-only register that can be written in 32-bit units
RSCANnTHLPCTRmL register is a write-only register that can be written in 16-bit units
RSCANnTHLPCTRmLL register is a write-only register that can be written in 8-bit units

**Address:** RSCANnTHLPCTRm: <RSCANn_base> + $0440_H$ + ($04_H$ × m)

RSCANnTHLPCTRmL: <RSCANn_base> + $0440_H$ + ($04_H$ × m)

RSCANnTHLPCTRmLL: <RSCANn_base> + $0440_H$ + ($04_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | THLPC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**Table 21.84 RSCANnTHLPCTRm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing, write the value after reset. |
| 7 to 0 | THLPC[7:0] | Transmit History List Pointer Control<br>Writing $FF_H$ to these bits moves the read pointer to the next unread data in the transmit history buffer. |

#### THLPC[7:0] Bits

When the THLPC[7:0] bits are set to $FF_H$, the read pointer moves to the next data in the transmit history buffer. At this time, the THLMC[4:0] (transmit history buffer unread data counter) value in the RSCANnTHLSTSm register is decremented. Write $FF_H$ to the THLPC[7:0] bits after reading from the RSCANnTHLACCm register.

When writing $FF_H$ to these bits, make sure that the THLE bit in the RSCANnTHLCCm register is set to 1 (transmit history buffer is used) and the THLEMP flag in the RSCANnTHLSTSm register is 0.

### 21.4.12.4 RSCANnTHLACCm — Transmit History Access Register (m = 0 to 5)

**Access:** RSCANnTHLACCm register is a read-only register that can be read in 32-bit units
RSCANnTHLACCmL, RSCANnTHLACCmH registers are read-only registers that can be read in 16-bit units
RSCANnTHLACCmLL, RSCANnTHLACCmLH, RSCANnTHLACCmHL, RSCANnTHLACCmHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCANnTHLACCm: <RSCANn_base> + $1800_H$ + ($04_H$ × m)

RSCANnTHLACCmL: <RSCANn_base> + $1800_H$ + ($04_H$ × m),
RSCANnTHLACCmH: <RSCANn_base> + $1802_H$ + ($04_H$ × m)

RSCANnTHLACCmLL: <RSCANn_base> + $1800_H$ + ($04_H$ × m),
RSCANnTHLACCmLH: <RSCANn_base> + $1801_H$ + ($04_H$ × m),
RSCANnTHLACCmHL: <RSCANn_base> + $1802_H$ + ($04_H$ × m),
RSCANnTHLACCmHH: <RSCANn_base> + $1803_H$ + ($04_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TID[7:0] | | | | | | | | — | BN[3:0] | | | | BT[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.85 RSCANnTHLACCm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | TMTS[15:0] | Transmit Timestamp Data<br>The timestamp data of stored data can be read. |
| 15 to 8 | TID[7:0] | Label Data<br>The label information of stored data can be read. |
| 7 | Reserved | When read, the value after reset is returned. |
| 6 to 3 | BN[3:0] | Buffer Number Data<br>The buffer number of transmit source (transmit buffer, transmit/receive FIFO buffer or transmit queue) can be read. |
| 2 to 0 | BT[2:0] | Buffer Type Data<br>b2 b1 b0<br>0 0 1: Transmit buffer<br>0 1 0: Transmit/receive FIFO buffer<br>1 0 0: Transmit queue |

#### TMTS[15:0] Bits

When the TMTSCE bit in the RSCANnGCFG register is 1, timestamp values in transmit history data stored in the transmit history buffer are displayed. When the TMTSCE bit is 0, these bits are always read as 0.

#### TID[7:0] Bits

These bits indicate the label information of transmit history data stored in the transmit history buffer.

#### BN[3:0] Bits

These bits indicate the transmit source buffer number in the transmit history data stored in the transmit history buffer.

**BT[2:0] Bits**

These bits indicate the type of the transmit source buffer in the transmit history data stored in the transmit history buffer.

## 21.4.13 Details of Test-Related Registers

### 21.4.13.1 RSCANnGTSTCFG — Global Test Configuration Register

**Access:** RSCANnGTSTCFG register can be read or written in 32-bit units
RSCANnGTSTCFGL, RSCANnGTSTCFGH registers can be read or written in 16-bit units
RSCANnGTSTCFGLL, RSCANnGTSTCFGHL registers can be read or written in 8-bit units

**Address:** RSCANnGTSTCFG: <RSCANn_base> + $0468_H$

RSCANnGTSTCFGL: <RSCANn_base> + $0468_H$,
RSCANnGTSTCFGH: <RSCANn_base> + $046A_H$

RSCANnGTSTCFGLL: <RSCANn_base> + $0468_H$,
RSCANnGTSTCFGHL: <RSCANn_base> + $046A_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | RTMPS[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | C5ICBCE | C4ICBCE | C3ICBCE | C2ICBCE | C1ICBCE | C0ICBCE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.86 RSCANnGTSTCFG Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 22 to 16 | RTMPS[6:0] | RAM Test Page Configuration<br>Set a value within the range of page 0 ($00_H$) to page 59 ($3B_H$). |
| 15 to 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 | C5ICBCE | CAN5 Inter-channel Communication Test Enable<br>0: CAN5 inter-channel communication test is disabled<br>1: CAN5 inter-channel communication test is enabled. |
| 4 | C4ICBCE | CAN4 Inter-channel Communication Test Enable<br>0: CAN4 inter-channel communication test is disabled<br>1: CAN4 inter-channel communication test is enabled. |
| 3 | C3ICBCE | CAN3 Inter-channel Communication Test Enable<br>0: CAN3 inter-channel communication test is disabled<br>1: CAN3 inter-channel communication test is enabled. |
| 2 | C2ICBCE | CAN2 Inter-channel Communication Test Enable<br>0: CAN2 inter-channel communication test is disabled<br>1: CAN2 inter-channel communication test is enabled. |
| 1 | C1ICBCE | CAN1 Inter-Channel Communication Test Enable<br>0: CAN1 inter-channel communication test is disabled.<br>1: CAN1 inter-channel communication test is enabled. |
| 0 | C0ICBCE | CAN0 Inter-Channel Communication Test Enable<br>0: CAN0 inter-channel communication test is disabled.<br>1: CAN0 inter-channel communication test is enabled. |

Modify the RSCANnGTSTCFG register only in global test mode.

#### RTMPS[6:0] Bits

These bits are used to set the RAM test target page number for RAM test. Set a value in the range of $00_H$ to $3B_H$, inclusive.

#### C5ICBCE Bit

Setting this bit to 1 enables the channel 5 inter-channel communication test.

This bit is cleared to 0 in global reset mode.

#### C4ICBCE Bit

Setting this bit to 1 enables the channel 4 inter-channel communication test.

This bit is cleared to 0 in global reset mode.

#### C3ICBCE Bit

Setting this bit to 1 enables the channel 3 inter-channel communication test.

This bit is cleared to 0 in global reset mode.

#### C2ICBCE Bit

Setting this bit to 1 enables the channel 2 inter-channel communication test.

This bit is cleared to 0 in global reset mode.

#### C1ICBCE Bit

Setting this bit to 1 enables the channel 1 inter-channel communication test.

This bit is cleared to 0 in global reset mode.

#### C0ICBCE Bit

Setting this bit to 1 enables the channel 0 inter-channel communication test.

This bit is cleared to 0 in global reset mode.

### 21.4.13.2 RSCANnGTSTCTR — Global Test Control Register

**Access:** RSCANnGTSTCTR register can be read or written in 32-bit units
RSCANnGTSTCTRL register can be read or written in 16-bit units
RSCANnGTSTCTRLL register can be read or written in 8-bit units

**Address:** RSCANnGTSTCTR: <RSCANn_base> + $046C_H$

RSCANnGTSTCTRL: <RSCANn_base> + $046C_H$

RSCANnGTSTCTRLL: <RSCANn_base> + $046C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | RTME | — | ICBCTME |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R/W |

**Table 21.87 RSCANnGTSTCTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 | RTME | RAM Test Enable<br>0: RAM test is disabled.<br>1: RAM test is enabled. |
| 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | ICBCTME | Communication Test between Channels Enable<br>0: Communication test between channels disabled<br>1: Communication test between channels enabled |

#### RTME Bit

Setting this bit to 1 enables the RAM test. Modify this bit only in global test mode.

This bit is cleared to 0 in global reset mode.

1. Set the GMDC[1:0] bits in the RSCANnGCTR register to $10_B$ (Global test mode).
2. Set the RTME bit to 1.
3. Check that the RTME bit is set to 1.

#### ICBCTME Bit

When this bit is set to 1, a communication test is enabled between the channels for which the CmICBCE bit (m = 0 to 5) in the RSCANnGTSTCFG register has been set to 1. Modify the ICBCTME bit only in global test mode.

This bit is cleared to 0 in global reset mode.

### 21.4.13.3 RSCANnGLOCKK — Global Lock Key Register

**Access:** RSCANnGLOCKK register is a write-only register that can be written in 32-bit units.
RSCANnGLOCKKL register is a write-only register that can be written in 16-bit units.

**Address:** RSCANnGLOCKK: <RSCANn_base> + $047C_H$

RSCANnGLOCKKL: <RSCANn_base> + $047C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LOCK[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 |

Note 1. Writing to these bits is effective only when the RS-CANFD module is in global test mode.

**Table 21.88 RSCANnGLOCKK Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When writing, write the value after reset. |
| 15 to 0 | LOCK[15:0] | Lock Key<br>These bits are key bits to release protection of test mode. |

The RSCANnGLOCKK register releases protection of special test bits and is write only.

For the protection release data, see **Section 21.11.4.2, Procedure for Releasing the Protection**.

#### LOCK[15:0] Bits

Writing the protection release data to the LOCK[15:0] bits in succession enables writing 1 to the RTME bit in the RSCANnGTSTCTR register.

After the protection has been released, writing to the I/O register area (<RSCANn_base> + $0000_H$ to <RSCANn_base> + $04FF_H$) of the CAN (except the RAM) enables the protection again.

Reading from the I/O register area of the CAN or reading from/writing to other areas does not enable the protection.

### 21.4.13.4 RSCANnRPGACCr — RAM Test Page Access Register (r = 0 to 63)

**Access:** RSCANnRPGACCr register can be read or written in 32-bit units
RSCANnRPGACCrL, RSCANnRPGACCrH registers can be read or written in 16-bit units
RSCANnRPGACCrLL, RSCANnRPGACCrLH, RSCANnRPGACCrHL, RSCANnRPGACCrHH registers can be read or written in 8-bit units

**Address:** RSCANnRPGACCr: <RSCANn_base> + $1900_H$ + ($04_H$ × r)

RSCANnRPGACCrL: <RSCANn_base> + $1900_H$ + ($04_H$ × r),
RSCANnRPGACCrH: <RSCANn_base> + $1902_H$ + ($04_H$ × r)

RSCANnRPGACCrLL: <RSCANn_base> + $1900_H$ + ($04_H$ × r),
RSCANnRPGACCrLH: <RSCANn_base> + $1901_H$ + ($04_H$ × r),
RSCANnRPGACCrHL: <RSCANn_base> + $1902_H$ + ($04_H$ × r),
RSCANnRPGACCrHH: <RSCANn_base> + $1903_H$ + ($04_H$ × r)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDTA[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDTA[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.89 RSCANnRPGACCr Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | RDTA [31:0] | RAM Data Test Access<br>Data can be read and written in RSCAN RAM. |

Modify the RSCANnRPGACCr register in global test mode with the RTME bit in the RSCANnGTSTCTR register set to 1 (RAM test is enabled).

The RSCANnRPGACCr register is readable and writable when the RTME bit is set to 1.

# 21.5 Registers (CAN FD Mode)

This section describes all registers to be used when the RS-CANFD is used in CAN FD mode.

## 21.5.1 List of Registers

The following tables list RS-CANFD registers to be used in CAN FD mode.

For details about <RSCANn_base>, see **Section 21.2.1, Register Base Address**.

**Table 21.90 Registers (1/3)**

| Module Name | Register | Symbol | Address | Guard Group |
|---|---|---|---|---|
| **Interface mode-related registers** | | | | |
| RSCFDn | Global interface mode select register | RSCFDnCFDGRMCFG | <RSCANn_base> + $04FC_H$ | RSCANn global |
| **Channel-related registers** | | | | |
| RSCFDn | Channel m nominal bit rate configuration register | RSCFDnCFDCmNCFG | <RSCANn_base> + $0000_H$ + ($10_H$ × m) | RSCANn Chm |
| | Channel m control register | RSCFDnCFDCmCTR | <RSCANn_base> + $0004_H$ + ($10_H$ × m) | RSCANn Chm |
| | Channel m status register | RSCFDnCFDCmSTS | <RSCANn_base> + $0008_H$ + ($10_H$ × m) | RSCANn Chm |
| | Channel m error flag register | RSCFDnCFDCmERFL | <RSCANn_base> + $000C_H$ + ($10_H$ × m) | RSCANn Chm |
| | Channel m data bit rate configuration register | RSCFDnCFDCmDCFG | <RSCANn_base> + $0500_H$ + ($20_H$ × m) | RSCANn Chm |
| | Channel m CAN FD configuration register | RSCFDnCFDCmFDCFG | <RSCANn_base> + $0504_H$ + ($20_H$ × m) | RSCANn Chm |
| | Channel m CAN FD control register | RSCFDnCFDCmFDCTR | <RSCANn_base> + $0508_H$ + ($20_H$ × m) | RSCANn Chm |
| | Channel m CAN FD status register | RSCFDnCFDCmFDSTS | <RSCANn_base> + $050C_H$ + ($20_H$ × m) | RSCANn Chm |
| | Channel m CAN FD CRC register | RSCFDnCFDCmFDCRC | <RSCANn_base> + $0510_H$ + ($20_H$ × m) | RSCANn Chm |
| **Global-related registers** | | | | |
| RSCFDn | Global configuration register | RSCFDnCFDGCFG | <RSCANn_base> + $0084_H$ | RSCANn global |
| | Global control register | RSCFDnCFDGCTR | <RSCANn_base> + $0088_H$ | RSCANn global |
| | Global status register | RSCFDnCFDGSTS | <RSCANn_base> + $008C_H$ | RSCANn global |
| | Global error flag register | RSCFDnCFDGERFL | <RSCANn_base> + $0090_H$ | RSCANn global |
| | Global timestamp counter register | RSCFDnCFDGTSC | <RSCANn_base> + $0094_H$ | RSCANn global |
| | Global TX Interrupt Status register 0 | RSCFDnCFDGTINTSTS0 | <RSCANn_base> + $0460_H$ | RSCANn global |
| | Global TX Interrupt Status register 1 | RSCFDnCFDGTINTSTS1 | <RSCANn_base> + $0464_H$ | RSCANn global |
| | Global FD configuration register | RSCFDnCFDGFDCFG | <RSCANn_base> + $0474_H$ | RSCANn global |
| **Receive rule-related registers** | | | | |
| RSCFDn | Receive Rule Entry Control register | RSCFDnCFDGAFLECTR | <RSCANn_base> + $0098_H$ | RSCANn global |
| | Receive Rule Configuration register 0 | RSCFDnCFDGAFLCFG0 | <RSCANn_base> + $009C_H$ | RSCANn global |
| | Receive Rule Configuration register 1 | RSCFDnCFDGAFLCFG1 | <RSCANn_base> + $00A0_H$ | RSCANn global |
| | Receive Rule ID register j | RSCFDnCFDGAFLIDj | <RSCANn_base> + $1000_H$ + ($10_H$ × j) | RSCANn global |
| | Receive Rule Mask register j | RSCFDnCFDGAFLMj | <RSCANn_base> + $1004_H$ + ($10_H$ × j) | RSCANn global |
| | Receive Rule Pointer 0 register j | RSCFDnCFDGAFLP0_j | <RSCANn_base> + $1008_H$ + ($10_H$ × j) | RSCANn global |
| | Receive Rule Pointer 1 register j | RSCFDnCFDGAFLP1_j | <RSCANn_base> + $100C_H$ + ($10_H$ × j) | RSCANn global |
| **Receive buffer-related registers** | | | | |
| RSCFDn | Receive Buffer Number register | RSCFDnCFDRMNB | <RSCANn_base> + $00A4_H$ | RSCANn global |
| | Receive Buffer New Data register y | RSCFDnCFDRMNDy | <RSCANn_base> + $00A8_H$ + ($04_H$ × y) | RSCANn global |
| | Receive Buffer ID register q | RSCFDnCFDRMIDq | <RSCANn_base> + $2000_H$ + ($20_H$ × q) | RSCANn global |

Table 21.90 Registers (2/3)

| Module Name | Register | Symbol | Address | Guard Group |
|---|---|---|---|---|
| RSCFDn | Receive Buffer Pointer register q | RSCFDnCFDRMPTRq | <RSCANn_base> + $2004_H$ + ($20_H$ × q) | RSCANn global |
| | Receive buffer CAN FD status register q | RSCFDnCFDRMFDSTSq | <RSCANn_base> + $2008_H$ + ($20_H$ × q) | RSCANn global |
| | Receive Buffer Data Field b register q | RSCFDnCFDRMDFb_q | <RSCANn_base> + $200C_H$ + ($04_H$ × b) + ($20_H$ × q) | RSCANn global |
| **Receive FIFO buffer-related registers** | | | | |
| RSCFDn | Receive FIFO Buffer Configuration and Control register x | RSCFDnCFDRFCCx | <RSCANn_base> + $00B8_H$ + ($04_H$ × x) | RSCANn global |
| | Receive FIFO Buffer Status register x | RSCFDnCFDRFSTSx | <RSCANn_base> + $00D8_H$ + ($04_H$ × x) | RSCANn global |
| | Receive FIFO Buffer Pointer Control register x | RSCFDnCFDRFPCTRx | <RSCANn_base> + $00F8_H$ + ($04_H$ × x) | RSCANn global |
| | Receive FIFO Buffer Access ID register x | RSCFDnCFDRFIDx | <RSCANn_base> + $3000_H$ + ($80_H$ × x) | RSCANn global |
| | Receive FIFO Buffer Access Pointer register x | RSCFDnCFDRFPTRx | <RSCANn_base> + $3004_H$ + ($80_H$ × x) | RSCANn global |
| | Receive FIFO CAN FD status register x | RSCFDnCFDRFFDSTSx | <RSCANn_base> + $3008_H$ + ($80_H$ × x) | RSCANn global |
| | Receive FIFO Buffer Access Data Field d register x | RSCFDnCFDRFDFd_x | <RSCANn_base> + $300C_H$ + ($04_H$ × d) + ($80_H$ × x) | RSCANn global |
| **Transmit/Receive FIFO buffer related registers** | | | | |
| RSCFDn | Transmit/receive FIFO Buffer Configuration and Control register k | RSCFDnCFDCFCCk | <RSCANn_base> + $0118_H$ + ($04_H$ × k) | RSCANn global |
| | Transmit/receive FIFO Buffer Status register k | RSCFDnCFDCFSTSk | <RSCANn_base> + $0178_H$ + ($04_H$ × k) | RSCANn global |
| | Transmit/receive FIFO Buffer Pointer Control register k | RSCFDnCFDCFPCTRk | <RSCANn_base> + $01D8_H$ + ($04_H$ × k) | RSCANn global |
| | Transmit/receive FIFO Buffer Access ID register k | RSCFDnCFDCFIDk | <RSCANn_base> + $3400_H$ + ($80_H$ × k) | RSCANn global |
| | Transmit/receive FIFO Buffer Access Pointer register k | RSCFDnCFDCFPTRk | <RSCANn_base> + $3404_H$ + ($80_H$ × k) | RSCANn global |
| | Transmit/receive FIFO CAN FD configuration/status register k | RSCFDnCFDCFFDCSTSk | <RSCANn_base> + $3408_H$ + ($80_H$ × k) | RSCANn global |
| | Transmit/receive FIFO Buffer Access Data Field d register k | RSCFDnCFDCFDFd_k | <RSCANn_base> + $340C_H$ + ($04_H$ × d) + ($80_H$ × k) | RSCANn global |
| **FIFO status-related registers** | | | | |
| RSCFDn | FIFO Empty Status register | RSCFDnCFDFESTS | <RSCANn_base> + $0238_H$ | RSCANn global |
| | FIFO Full Status register | RSCFDnCFDFFSTS | <RSCANn_base> + $023C_H$ | RSCANn global |
| | FIFO Message Lost Status register | RSCFDnCFDFMSTS | <RSCANn_base> + $0240_H$ | RSCANn global |
| | Receive FIFO Buffer Interrupt Flag Status register | RSCFDnCFDRFISTS | <RSCANn_base> + $0244_H$ | RSCANn global |
| | Transmit/receive FIFO Buffer Receive Interrupt Flag Status register | RSCFDnCFDCFRISTS | <RSCANn_base> + $0248_H$ | RSCANn global |
| | Transmit/receive FIFO Buffer Transmit Interrupt Flag Status register | RSCFDnCFDCFTISTS | <RSCANn_base> + $024C_H$ | RSCANn global |
| **FIFO DMA-related registers** | | | | |
| RSCFDn | DMA enable register | RSCFDnCFDCDTCT | <RSCANn_base> + $0490_H$ | RSCANn global |
| | DMA status register | RSCFDnCFDCDTSTS | <RSCANn_base> + $0494_H$ | RSCANn global |
| **Transmit buffer-related registers** | | | | |
| RSCFDn | Transmit Buffer Control register p | RSCFDnCFDTMCp | <RSCANn_base> + $0250_H$ + ($01_H$ × p) | RSCANn global |
| | Transmit Buffer Status register p | RSCFDnCFDTMSTSp | <RSCANn_base> + $02D0_H$ + ($01_H$ × p) | RSCANn global |
| | Transmit Buffer ID register p | RSCFDnCFDTMIDp | <RSCANn_base> + $4000_H$ + ($20_H$ × p) | RSCANn global |
| | Transmit Buffer Pointer register p | RSCFDnCFDTMPTRp | <RSCANn_base> + $4004_H$ + ($20_H$ × p) | RSCANn global |
| | Transmit buffer CAN FD configuration register p | RSCFDnCFDTMFDCTRp | <RSCANn_base> + $4008_H$ + ($20_H$ × p) | RSCANn global |
| | Transmit Buffer Data Field b register p | RSCFDnCFDTMDFb_p | <RSCANn_base> + $400C_H$ + ($04_H$ × b) + ($20_H$ × p) | RSCANn global |
| | Transmit Buffer Interrupt Enable Configuration register y | RSCFDnCFDTMIECy | <RSCANn_base> + $0390_H$ + ($04_H$ × y) | RSCANn global |

Table 21.90 Registers (3/3)

| Module Name | Register | Symbol | Address | Guard Group |
|---|---|---|---|---|
| **Transmit buffer status-related registers** | | | | |
| RSCFDn | Transmit Buffer Transmit Request Status register y | RSCFDnCFDTMTRSTSy | <RSCANn_base> + $0350_H$ + ($04_H$ × y) | RSCANn global |
| | Transmit Buffer Transmit Abort Request Status register y | RSCFDnCFDTMTARSTSy | <RSCANn_base> + $0360_H$ + ($04_H$ × y) | RSCANn global |
| | Transmit Buffer Transmit Complete Status register y | RSCFDnCFDTMTCSTSy | <RSCANn_base> + $0370_H$ + ($04_H$ × y) | RSCANn global |
| | Transmit Buffer Transmit Abort Status register y | RSCFDnCFDTMTASTSy | <RSCANn_base> + $0380_H$ + ($04_H$ × y) | RSCANn global |
| **Transmit queue-related registers** | | | | |
| RSCFDn | Transmit Queue Configuration and Control register m | RSCFDnCFDTXQCCm | <RSCANn_base> + $03A0_H$ + ($04_H$ × m) | RSCANn Chm |
| | Transmit Queue Status register m | RSCFDnCFDTXQSTSm | <RSCANn_base> + $03C0_H$ + ($04_H$ × m) | RSCANn Chm |
| | Transmit Queue Pointer Control register m | RSCFDnCFDTXQPCTRm | <RSCANn_base> + $03E0_H$ + ($04_H$ × m) | RSCANn Chm |
| **Transmit history-related registers** | | | | |
| RSCFDn | Transmit History Configuration and Control register m | RSCFDnCFDTHLCCm | <RSCANn_base> + $0400_H$ + ($04_H$ × m) | RSCANn Chm |
| | Transmit History Status register m | RSCFDnCFDTHLSTSm | <RSCANn_base> + $0420_H$ + ($04_H$ × m) | RSCANn Chm |
| | Transmit History Pointer Control register m | RSCFDnCFDTHLPCTRm | <RSCANn_base> + $0440_H$ + ($04_H$ × m) | RSCANn Chm |
| | Transmit History Access register m | RSCFDnCFDTHLACCm | <RSCANn_base> + $6000_H$ + ($04_H$ × m) | RSCANn Chm |
| **Test-related registers** | | | | |
| RSCFDn | Global Test Configuration register | RSCFDnCFDGTSTCFG | <RSCANn_base> + $0468_H$ | RSCANn global |
| | Global Test Control register | RSCFDnCFDGTSTCTR | <RSCANn_base> + $046C_H$ | RSCANn global |
| | Global Lock Key register | RSCFDnCFDGLOCKK | <RSCANn_base> + $047C_H$ | RSCANn global |
| | RAM Test Page Access register r | RSCFDnCFDRPGACCr | <RSCANn_base> + $6400_H$ + ($04_H$ × r) | RSCANn global |

**Note:** For details on the guard group, see **Section 33, Functional Safety**.

**Table 21.91 Transmit Buffer p Allocated to Each Channel**

| | CANm |
|---|---|
| Transmit buffer p | Transmit buffer 16 × m + 0 |
| | Transmit buffer 16 × m + 1 |
| | Transmit buffer 16 × m + 2 |
| | Transmit buffer 16 × m + 3 |
| | Transmit buffer 16 × m + 4 |
| | Transmit buffer 16 × m + 5 |
| | Transmit buffer 16 × m + 6 |
| | Transmit buffer 16 × m + 7 |
| | Transmit buffer 16 × m + 8 |
| | Transmit buffer 16 × m + 9 |
| | Transmit buffer 16 × m + 10 |
| | Transmit buffer 16 × m + 11 |
| | Transmit buffer 16 × m + 12 |
| | Transmit buffer 16 × m + 13 |
| | Transmit buffer 16 × m + 14 |
| | Transmit buffer 16 × m + 15 |

**Table 21.92 Transmit/Receive FIFO Buffer k Allocated to Each Channel**

| | CANm |
|---|---|
| Transmit/receive FIFO buffer k | Transmit/receive FIFO buffer 3 × m + 0 |
| | Transmit/receive FIFO buffer 3 × m + 1 |
| | Transmit/receive FIFO buffer 3 × m + 2 |

**Table 21.93 Transmit Buffer p Linked to the Transmit/Receive FIFO Buffer by the Setting of Bits CFTML[3:0]**

| Setting of Bits CFTML[3:0] | Transmit Buffer p Linked to the Transmit/Receive FIFO Buffer |
|---|---|
| $0000_B$ | Transmit buffer 16 × m + 0 |
| $0001_B$ | Transmit buffer 16 × m + 1 |
| $0010_B$ | Transmit buffer 16 × m + 2 |
| $0011_B$ | Transmit buffer 16 × m + 3 |
| $0100_B$ | Transmit buffer 16 × m + 4 |
| $0101_B$ | Transmit buffer 16 × m + 5 |
| $0110_B$ | Transmit buffer 16 × m + 6 |
| $0111_B$ | Transmit buffer 16 × m + 7 |
| $1000_B$ | Transmit buffer 16 × m + 8 |
| $1001_B$ | Transmit buffer 16 × m + 9 |
| $1010_B$ | Transmit buffer 16 × m + 10 |
| $1011_B$ | Transmit buffer 16 × m + 11 |
| $1100_B$ | Transmit buffer 16 × m + 12 |
| $1101_B$ | Transmit buffer 16 × m + 13 |
| $1110_B$ | Transmit buffer 16 × m + 14 |
| $1111_B$ | Transmit buffer 16 × m + 15 |

**Table 21.94 Transmit Buffer p Allocated to the Transmit Queue of Each Channel**

| Setting of Bits TXQDC[3:0] | Transmit Buffer p Allocated to the Transmit Queue |
|---|---|
| $0000_B$ | Setting prohibited |
| $0001_B$ | Setting prohibited |
| $0010_B$ | Transmit buffer 16 × m + 15 to 16 × m + 13 |
| $0011_B$ | Transmit buffer 16 × m + 15 to 16 × m + 12 |
| $0100_B$ | Transmit buffer 16 × m + 15 to 16 × m + 11 |
| $0101_B$ | Transmit buffer 16 × m + 15 to 16 × m + 10 |
| $0110_B$ | Transmit buffer 16 × m + 15 to 16 × m + 9 |
| $0111_B$ | Transmit buffer 16 × m + 15 to 16 × m + 8 |
| $1000_B$ | Transmit buffer 16 × m + 15 to 16 × m + 7 |
| $1001_B$ | Transmit buffer 16 × m + 15 to 16 × m + 6 |
| $1010_B$ | Transmit buffer 16 × m + 15 to 16 × m + 5 |
| $1011_B$ | Transmit buffer 16 × m + 15 to 16 × m + 4 |
| $1100_B$ | Transmit buffer 16 × m + 15 to 16 × m + 3 |
| $1101_B$ | Transmit buffer 16 × m + 15 to 16 × m + 2 |
| $1110_B$ | Transmit buffer 16 × m + 15 to 16 × m + 1 |
| $1111_B$ | Transmit buffer 16 × m + 15 to 16 × m + 0 |

## 21.5.2 Details of Interface Mode-Related Registers

### 21.5.2.1 RSCFDnCFDGRMCFG — Global Interface Mode Select Register

**Access:** RSCFDnCFDGRMCFG register can be read or written in 32-bit units
RSCFDnCFDGRMCFGL register can be read or written in 16-bit units
RSCFDnCFDGRMCFGLL register can be read or written in 8-bit units

**Address:** RSCFDnCFDGRMCFG: <RSCANn_base> + $04FC_H$

RSCFDnCFDGRMCFGL: <RSCANn_base> + $04FC_H$

RSCFDnCFDGRMCFGLL: <RSCANn_base> + $04FC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RCMC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 21.95 RSCFDnCFDGRMCFG Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | RCMC | Interface Mode Select<br>0: Classical CAN mode<br>1: CAN FD mode (PREMIUM only) |

**Note:** CAN FD mode for PREMIUM only. Even if RCMC bit is set to 1, CANFD mode does not work in ECO and ADVANCED products. Set RCMC = 0 in ECO and ADVANCED products.

Modify the RSCFDnCFDGRMCFG register only in global reset mode. Before setting other RS-CANFD registers, set this register.

#### RCMC Bit

Setting this bit to 0 makes classical CAN mode available. To switch CAN FD mode to classical CAN mode, set the value after reset to all registers and bits allocated to the register map of CAN FD mode and then modify the RSCFDnCFDGRMCFG register.

## 21.5.3 Details of Channel-Related Registers

### 21.5.3.1 RSCFDnCFDCmNCFG — Channel Nominal Bit Rate Configuration Register (m = 0 to 5)

**Access:** RSCFDnCFDCmNCFG register can be read or written in 32-bit units
RSCFDnCFDCmNCFGL, RSCFDnCFDCmNCFGH registers can be read or written in 16-bit units
RSCFDnCFDCmNCFGLL, RSCFDnCFDCmNCFGLH, RSCFDnCFDCmNCFGHL,
RSCFDnCFDCmNCFGHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDCmNCFG: <RSCANn_base> + $0000_H$ + ($10_H$ × m)

RSCFDnCFDCmNCFGL: <RSCANn_base> + $0000_H$ + ($10_H$ × m),
RSCFDnCFDCmNCFGH: <RSCANn_base> + $0002_H$ + ($10_H$ × m)

RSCFDnCFDCmNCFGLL: <RSCANn_base> + $0000_H$ + ($10_H$ × m),
RSCFDnCFDCmNCFGLH: <RSCANn_base> + $0001_H$ + ($10_H$ × m),
RSCFDnCFDCmNCFGHL: <RSCANn_base> + $0002_H$ + ($10_H$ × m),
RSCFDnCFDCmNCFGHH: <RSCANn_base> + $0003_H$ + ($10_H$ × m)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | NTSEG2[4:0] | | | | | — | NTSEG1[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NSJW[4:0] | | | | | — | NBRP[9:0] | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.96 RSCFDnCFDCmNCFG Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 28 to 24 | NTSEG2[4:0] | Nominal Bit Rate Time Segment 2 Control<br>b28 b27 b26 b25 b24<br>0 0 0 0 0: Setting prohibited<br>0 0 0 0 1: 2 Tq<br>:<br>:<br>1 1 1 1 0: 31 Tq<br>1 1 1 1 1: 32 Tq |
| 23 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 22 to 16 | NTSEG1[6:0] | Nominal Bit Rate Time Segment 1 Control<br>b22 b21 b20 b19 b18 b17 b16<br>0 0 0 0 0 0 0: Setting prohibited<br>0 0 0 0 0 0 1: Setting prohibited<br>0 0 0 0 0 1 0: Setting prohibited<br>0 0 0 0 0 1 1: 4 Tq<br>:<br>:<br>1 1 1 1 1 1 0: 127 Tq<br>1 1 1 1 1 1 1: 128 Tq |

Table 21.96 RSCFDnCFDCmNCFG Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 11 | NSJW[4:0] | Nominal Bit Rate Resynchronization Jump Width Control<br>b15 b14 b13 b12 b11<br>0 0 0 0 0: 1 Tq<br>0 0 0 0 1: 2 Tq<br>0 0 0 1 0: 3 Tq<br>:<br>:<br>1 1 1 1 0: 31 Tq<br>1 1 1 1 1: 32 Tq |
| 10 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 9 to 0 | NBRP[9:0] | Nominal Bit Rate Prescaler Division Ratio Setting<br>When the set value = P (0 to 1023), the nominal bit rate prescaler divides fCAN by (P + 1). |

Modify the RSCFDnCFDCmNCFG register in channel reset mode or channel halt mode. Set this register in channel reset mode, and then transition to channel communication mode or channel halt mode. For the description and settings of bit timing parameters, see **Section 21.11.1, Initial Settings**.

#### NTSEG2[4:0] Bits

These bits are used to specify a Tq value for the length of phase segment 2 (PHASE_SEG2) of nominal bit rate.

Allowed values are 2 Tq to 32 Tq, inclusive.

Set a value smaller than the value of the NTSEG1[6:0] bits.

#### NTSEG1[6:0] Bits

These bits are used to specify the total length of propagation segment (PROP_SEG) and phase segment 1 (PHASE_SEG1) of nominal bit rate as a Tq value.

A value of 4 to 128 Tq can be set.

#### NSJW[4:0] Bits

These bits are used to specify the resynchronization jump width of nominal bit rate as a Tq value. A value of 1 to 32 Tq can be set. Specify a value equal to or smaller than the NTSEG2[4:0] value.

#### NBRP[9:0] Bits

The clock obtained by dividing the CAN clock (fCAN) by the nominal bit rate prescaler ((NBRP[9:0]) + 1) becomes CANmTq(N) clock (fCANTQ(N)m). One clock of the CANmTq(N) clock becomes one Time Quantum (Tq).

The NBRP[9:0] bit value and the DBRP[7:0] bit value in RSCFDnCFDCmDCFG register should be equal and the two bit rate values be different according to the respective segment values.

### 21.5.3.2 RSCFDnCFDCmCTR — Channel Control Register (m = 0 to 5)

**Access:** RSCFDnCFDCmCTR register can be read or written in 32-bit units
RSCFDnCFDCmCTRL, RSCFDnCFDCmCTRH registers can be read or written in 16-bit units
RSCFDnCFDCmCTRLL, RSCFDnCFDCmCTRLH, RSCFDnCFDCmCTRHL, RSCFDnCFDCmCTRHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDCmCTR: <RSCANn_base> + $0004_H$ + ($10_H$ × m)

RSCFDnCFDCmCTRL: <RSCANn_base> + $0004_H$ + ($10_H$ × m),
RSCFDnCFDCmCTRH: <RSCANn_base> + $0006_H$ + ($10_H$ × m)

RSCFDnCFDCmCTRLL: <RSCANn_base> + $0004_H$ + ($10_H$ × m),
RSCFDnCFDCmCTRLH: <RSCANn_base> + $0005_H$ + ($10_H$ × m),
RSCFDnCFDCmCTRHL: <RSCANn_base> + $0006_H$ + ($10_H$ × m),
RSCFDnCFDCmCTRHH: <RSCANn_base> + $0007_H$ + ($10_H$ × m)

**Value after reset:** $0000\ 0005_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ROM | CRCT | — | — | — | CTMS[1:0] | | CTME | ERRD | BOM[1:0] | | — | TDCVFIE | SOCOIE | EOCOIE | TAIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ALIE | BLIE | OLIE | BORIE | BOEIE | EPIE | EWIE | BEIE | — | — | — | — | RTBO | CSLPR | CHMDC[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 21.97 RSCFDnCFDCmCTR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | ROM | Restricted Operation Mode Enable<br>0: Restricted operation mode is disabled.<br>1: Restricted operation mode is enabled. |
| 30 | CRCT | CRC Error Test Enable<br>0: The first bit of the reception ID field is not inverted.<br>1: The first bit of the reception ID field is inverted. |
| 29 to 27 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 26, 25 | CTMS[1:0] | Communication Test Mode Select<br>b26 b25<br>0 0: Standard test mode<br>0 1: Listen-only mode<br>1 0: Self-test mode 0 (external loopback mode)<br>1 1: Self-test mode 1 (internal loopback mode) |
| 24 | CTME | Communication Test Mode Enable<br>0: Communication test mode is disabled.<br>1: Communication test mode is enabled. |
| 23 | ERRD | Error Display Mode Select<br>0: Error flags are displayed only for the first error information after bits 14 to 8 in the RSCFDnCFDCmERFL register are all cleared.<br>1: Error flags for all error information are displayed. |
| 22, 21 | BOM[1:0] | Bus Off Recovery Mode Select<br>b22 b21<br>0 0: ISO11898-1 compliant<br>0 1: Entry to channel halt mode automatically at bus-off entry<br>1 0: Entry to channel halt mode automatically at bus-off end<br>1 1: Entry to channel halt mode (in bus-off state) by program request |
| 20 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

Table 21.97 RSCFDnCFDCmCTR Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 19 | TDCVFIE | Transmitter Delay Compensation Violation Interrupt Enable<br>0: A transmitter delay compensation violation interrupt is disabled.<br>1: A transmitter delay compensation violation interrupt is enabled. |
| 18 | SOCOIE | Successful Occurrence Counter Overflow Interrupt Enable<br>0: A successful occurrence counter overflow interrupt is disabled.<br>1: A successful occurrence counter overflow interrupt is enabled. |
| 17 | EOCOIE | Error Occurrence Counter Overflow Interrupt Enable<br>0: An error occurrence counter overflow interrupt is disabled.<br>1: An error occurrence counter overflow interrupt is enabled. |
| 16 | TAIE | Transmit Abort Interrupt Enable<br>0: Transmit abort interrupt is disabled.<br>1: Transmit abort interrupt is enabled. |
| 15 | ALIE | Arbitration Lost Interrupt Enable<br>0: Arbitration lost interrupt is disabled.<br>1: Arbitration lost interrupt is enabled. |
| 14 | BLIE | Bus Lock Interrupt Enable<br>0: Bus lock interrupt is disabled.<br>1: Bus lock interrupt is enabled. |
| 13 | OLIE | Overload Frame Transmit Interrupt Enable<br>0: Overload frame transmit interrupt is disabled.<br>1: Overload frame transmit interrupt is enabled. |
| 12 | BORIE | Bus Off Recovery Interrupt Enable<br>0: Bus off recovery interrupt is disabled.<br>1: Bus off recovery interrupt is enabled. |
| 11 | BOEIE | Bus Off Entry Interrupt Enable<br>0: Bus off entry interrupt is disabled.<br>1: Bus off entry interrupt is enabled. |
| 10 | EPIE | Error Passive Interrupt Enable<br>0: Error passive interrupt is disabled.<br>1: Error passive interrupt is enabled. |
| 9 | EWIE | Error Warning Interrupt Enable<br>0: Error warning interrupt is disabled.<br>1: Error warning interrupt is enabled. |
| 8 | BEIE | Bus Error Interrupt Enable<br>0: Bus error interrupt is disabled.<br>1: Bus error interrupt is enabled. |
| 7 to 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | RTBO | Forcible Return from Bus-off<br>When this bit is set to 1, forcible return from the bus off state is made. This bit is always read as 0. |
| 2 | CSLPR | Channel Stop Mode<br>0: Other than channel stop mode<br>1: Channel stop mode |
| 1, 0 | CHMDC[1:0] | Mode Select<br>b1 b0<br>0 0: Channel communication mode<br>0 1: Channel reset mode<br>1 0: Channel halt mode<br>1 1: Setting prohibited |

### ROM Bit

When the ROM bit and the CTME bit in the RSCFDnCFDCmCTR register are set to 1, restricted operation mode is enabled. Use the restricted operation mode only when the CTMS[1:0] value in the RSCFDnCFDCmCTR register is $00_B$ (standard test mode). Modify this bit only in channel halt mode. This bit is always 0 in channel reset mode.

#### CRCT Bit

This bit is used to test the CRC generation circuit in the RS-CANFD module. Setting this bit to 1 inverts the first bit of the ID field when a message is received. With this inversion of bit, the CRC calculation result does not match the normal CRC value of the received frame, which can detect a CRC error (the CERR bit in the RSCFDnCFDCmERFL register is 1). When using this function, note the following.

- This function is available while the CTME bit in the RSCFDnCFDCmCTR register is 1 (communication test mode enabled).
- This function cannot communicate with other CAN nodes. Use this function for inter-channel communication test (the CmICBCE bit in the RSCFDnCFDGTSTCFG register is 1).
- Bit inversion in the ID field may cause bit stuffing rule violation. In that case, no CRC error is detected but a stuff error is detected.

Modify this bit only in channel halt mode. This bit is always 0 in channel reset mode.

#### CTMS[1:0] Bits

These bits are used to select a communication test mode. Modify these bits in channel halt mode only. These bits are set to 0 in channel reset mode.

#### CTME Bit

Setting this bit to 1 enables communication test mode. Modify this bit in channel halt mode. This bit is set to 0 in channel reset mode.

#### ERRD Bit

This bit is used to control the display mode of bits 14 to 8 in the RSCFDnCFDCmERFL register. When this bit is cleared to 0, if any error is detected while the flags of bits 14 to 8 in the RSCFDnCFDCmERFL register are all 0, only the flags of the first error event are set to 1. If two or more errors occur in the first error event, all the flags of the detected errors are set to 1.

When this bit is set to 1, all the flags of errors that have occurred are set to 1 regardless of the error occurrence order.

Modify this bit only in channel reset mode or channel halt mode.

#### BOM[1:0] Bits

These bits are used to select the bus off recovery mode of the RS-CANFD module.

When the BOM[1:0] bits are set to $00_B$, return from the bus off state to the error active state is compliant with the CAN specifications. That is, the RS-CANFD module reenters the CAN communication (error active state) after 11 consecutive recessive bits are detected 128 times. A bus off recovery interrupt request is generated at the time of return from the bus off state. Even if the CHMDC[1:0] bits are set to $10_B$ (channel halt mode) before recessive bits are detected 128 times, the RS-CANFD module does not transition to channel halt mode until recessive bits are detected 128 times.

When the RS-CANFD module reaches the bus off state when the BOM[1:0] bits are set to $01_B$, the CHMDC[1:0] bits in the RSCFDnCFDCmCTR register (m = 0 to 5) are set to $10_B$ and the RS-CANFD module transitions to channel halt mode. No bus off recovery interrupt request is generated and the TEC[7:0] and REC[7:0] bits in the RSCFDnCFDCmSTS register are cleared to $00_H$.

When the RS-CANFD module reaches the bus off state when the BOM[1:0] bits are set to $10_B$, the CHMDC[1:0] bits are set to $10_B$ and the RS-CANFD module transitions to channel halt mode after return from the bus off state (11 consecutive recessive bits are detected 128 times). A bus off recovery interrupt request is generated at the time of return from the bus off state and the TEC[7:0] and REC[7:0] bits are cleared to $00_H$.

When the BOM[1:0] bits are set to $11_B$ and the CHMDC[1:0] bits are set to $10_B$ while the RS-CANFD module is in the bus off state, the RS-CANFD module transitions to channel halt mode. No bus off recovery interrupt request is generated at the time of return from the bus off state and the TEC[7:0] and REC[7:0] bits are cleared to $00_H$. However, if 11 consecutive recessive bits are detected 128 times and the RS-CANFD module has recovered to the error active state from the bus off state before the CHMDC[1:0] bits are set to $10_B$, a bus off recovery interrupt request is generated.

If a program writes to the CHMDC[1:0] bit at the same time as the RS-CANFD module transition to channel halt mode (at bus off entry when the BOM[1:0] bits are 01B or at bus off end when the BOM[1:0] bits are $10_B$), the program's writing takes precedence. Modify the BOM[1:0] bits only in channel reset mode.

#### TDCVFIE Bit

When the TDCVF flag in the RSCFDnCFDCmFDSTS register is set to 1 after the TDCVFIE bit is set to 1, an interrupt request occurs. Modify this bit only in channel reset mode.

#### SOCOIE Bit

When the SOCO flag in the RSCFDnCFDCmFDSTS register is set to 1 after the SOCOIE bit is set to 1, an interrupt request occurs. Modify this bit only in channel reset mode.

#### EOCOIE Bit

When the EOCO flag in the RSCFDnCFDCmFDSTS register is set to 1 after the EOCOIE bit is set to 1, an interrupt request occurs. Modify this bit only in channel reset mode.

#### TAIE Bit

When transmit abort of the transmit buffer is completed with the TAIE bit set to 1, an interrupt request is generated. Modify this bit only in channel reset mode.

#### ALIE Bit

When the ALF flag in the RSCFDnCFDCmERFL register is set to 1 with the ALIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### BLIE Bit

When the BLF flag in the RSCFDnCFDCmERFL register is set to 1 with the BLIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### OLIE Bit

When the OVLF flag in the RSCFDnCFDCmERFL register is set to 1 with the OLIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### BORIE Bit

When the BORF flag in the RSCFDnCFDCmERFL register is set to 1 with the BORIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### BOEIE Bit

When the BOEF flag in the RSCFDnCFDCmERFL register is set to 1 with the BOEIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### EPIE Bit

When the EPF flag in the RSCFDnCFDCmERFL register is set to 1 with the EPIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### EWIE Bit

When the EWF flag in the RSCFDnCFDCmERFL register is set to 1 with the EWIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### BEIE Bit

When the BEF flag in the RSCFDnCFDCmERFL register is set to 1 with the BEIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### RTBO Bit

Setting this bit to 1 in the bus off state forcibly returns the state from the bus off state to the error active state. This bit is automatically cleared to 0. Setting this bit to 1 clears the TEC[7:0] and REC[7:0] bits in the RSCFDnCFDCmSTS register to $00_H$ and also clears the BOSTS flag in the RSCFDnCFDCmSTS register to 0 (not in bus off state). The other registers remain unchanged. No bus off recovery interrupt request is generated upon return from the bus off state in this case. Use this bit only when the BOM[1:0] bits in the RSCFDnCFDCmCTR register are $00_B$ (ISO11898-1 compliant).

A delay of up to 1 CAN bit time occurs after the RTBO bit is set to 1 until the RS-CANFD module transitions to the error active state. Set this bit to 1 in channel communication mode.

#### CSLPR Bit

Setting this bit to 1 places the channel into channel stop mode.
Clearing this bit to 0 makes the channel exit channel stop mode.
This bit should not be modified in channel communication mode or channel halt mode.

#### CHMDC[1:0] Bits

These bits are used to select a channel mode (channel communication mode, channel reset mode, or channel halt mode). For details, see **Section 21.7.2, Channel Modes**. Setting the CSLPR bit to 1 in channel reset mode allows transition to channel stop mode. Do not set the CHMDC[1:0] bits to $11_B$. When the CAN module has automatically transitioned to channel halt mode based on the setting of the BOM[1:0] bits, the CHMDC[1:0] bits automatically become $10_B$.

### 21.5.3.3 RSCFDnCFDCmSTS — Channel Status Register (m = 0 to 5)

**Access:** RSCFDnCFDCmSTS register can be read or written in 32-bit units
RSCFDnCFDCmSTSL register can be read or written in 16-bit units
RSCFDnCFDCmSTSH register is a read-only register that can be read in 16-bit units
RSCFDnCFDCmSTSLL register is a read-only register that can be read in 8-bit units
RSCFDnCFDCmSTSLH register can be read or written in 8-bit units
RSCFDnCFDCmSTSHL, RSCFDnCFDCmSTSHH registers are the read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDCmSTS: <RSCANn_base> + $0008_H$ + ($10_H$ × m)

RSCFDnCFDCmSTSL: <RSCANn_base> + $0008_H$ + ($10_H$ × m),
RSCFDnCFDCmSTSH: <RSCANn_base> + $000A_H$ + ($10_H$ × m)

RSCFDnCFDCmSTSLL: <RSCANn_base> + $0008_H$ + ($10_H$ × m),
RSCFDnCFDCmSTSLH: <RSCANn_base> + $0009_H$ + ($10_H$ × m),
RSCFDnCFDCmSTSHL: <RSCANn_base> + $000A_H$ + ($10_H$ × m),
RSCFDnCFDCmSTSHH: <RSCANn_base> + $000B_H$ + ($10_H$ × m)

**Value after reset:** $0000\ 0005_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEC[7:0] | | | | | | | | REC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | ESIF | COMSTS | RECSTS | TRMSTS | BOSTS | EPSTS | CSLPSTS | CHLTSTS | CRSTSTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R/W*1 | R | R | R | R | R | R | R | R |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.98 RSCFDnCFDCmSTS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | TEC[7:0] | The transmit error counter (TEC) can be read. |
| 23 to 16 | REC[7:0] | The receive error counter (REC) can be read. |
| 15 to 9 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 8 | ESIF | Error State Indication Flag<br>0: No CAN FD message whose ESI bit is recessive has been received.<br>1: At least one CAN FD message whose ESI bit is recessive has been received. |
| 7 | COMSTS | Communication Status Flag<br>0: Communication is not ready.<br>1: Communication is ready. |
| 6 | RECSTS | Receive Status Flag<br>0: Bus idle, in transmission or bus off state<br>1: In reception |
| 5 | TRMSTS | Transmit Status Flag<br>0: Bus idle or in reception<br>1: In transmission or bus off state |
| 4 | BOSTS | Bus Off Status Flag<br>0: Not in bus off state<br>1: In bus off state |
| 3 | EPSTS | Error Passive Status Flag<br>0: Not in error passive state<br>1: In error passive state |

Table 21.98 RSCFDnCFDCmSTS Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 2 | CSLPSTS | Channel Stop Status Flag<br>0: Not in channel stop mode<br>1: In channel stop mode |
| 1 | CHLTSTS | Channel Halt Status Flag<br>0: Not in channel halt mode<br>1: In channel halt mode |
| 0 | CRSTSTS | Channel Reset Status Flag<br>0: Not in channel reset mode<br>1: In channel reset mode |

### TEC[7:0] Bits

These bits contain the transmit error counter value. For transmit error counter increment/decrement conditions, see the CAN specification (ISO11898-1).

These bits are cleared to 0 in channel reset mode.

### REC[7:0] Bits

These bits contain the receive error counter value. For receive error counter increment/decrement conditions, see the CAN specifications (ISO11898-1).

These bits are cleared to 0 in channel reset mode.

### ESIF Flag

When the recessive ESI bit is detected in a successfully received message, this flag is set to 1. In loopback mode or mirror mode, the own transmission message is regarded as a received message. To clear this flag to 0, write 0 to this bit by the program. This bit cannot be set to 1 by the program. If the flag setting (to 1) timing matches the writing 0 (by the program) timing, this flag is set to 1.

This flag is 0 in channel reset mode.

### COMSTS Flag

This bit indicates that communication is ready.

This flag becomes 1 when the CAN module has detected 11 consecutive recessive bits after it has transitioned from channel reset mode or channel halt mode to channel communication mode. This flag is cleared to 0 in channel reset mode or channel halt mode.

### RECSTS Flag

This flag is set to 1 when reception has started, and is cleared to 0 when the bus has become idle or transmission has started.

### TRMSTS Flag

This flag is set to 1 when transmission has started, and is cleared to 0 when the bus has become idle or reception has started. This flag remains 1 in the bus off state.

### BOSTS Flag

This flag is set to 1 when the bus off state (TEC[7:0] > 255) is entered. It is cleared to 0 when the CAN module has exited the bus off state.

### EPSTS Flag

This flag is set to 1 when the RS-CANFD module has entered the error passive state ((128 ≤ TEC[7:0] ≤ 255) or (128 ≤ REC[7:0])). It is cleared to 0 when the RS-CANFD module has exited the error passive state or has entered channel reset mode.

### CSLPSTS Flag

This flag is set to 1 when the CAN module has transitioned to channel stop mode, and is cleared to 0 when the CAN module has returned from channel stop mode.

### CHLTSTS Flag

This flag is set to 1 when the CAN module has transitioned to channel halt mode, and is cleared to 0 when the CAN module has returned from channel halt mode.

### CRSTSTS Flag

This flag is set to 1 when the CAN module has transitioned to channel reset mode, and is cleared to 0 when the CAN module has transitioned to channel communication mode or channel halt mode. This flag remains 1 when the CAN module transitions from channel reset mode to channel stop mode.

### 21.5.3.4 RSCFDnCFDCmERFL — Channel Error Flag Register (m = 0 to 5)

**Access:** RSCFDnCFDCmERFL register can be read or written in 32-bit units
RSCFDnCFDCmERFLL register can be read or written in 16-bit units
RSCFDnCFDCmERFLH register is a read-only register that can be read in 16-bit units
RSCFDnCFDCmERFLLL, RSCFDnCFDCmERFLLH registers can be read or written in 8-bit units
RSCFDnCFDCmERFLHL, RSCFDnCFDCmERFLHH registers are the read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDCmERFL: <RSCANn_base> + $000C_H$ + ($10_H$ × m)

RSCFDnCFDCmERFLL: <RSCANn_base> + $000C_H$ + ($10_H$ × m),
RSCFDnCFDCmERFLH: <RSCANn_base> + $000E_H$ + ($10_H$ × m)

RSCFDnCFDCmERFLLL: <RSCANn_base> + $000C_H$ + ($10_H$ × m),
RSCFDnCFDCmERFLLH: <RSCANn_base> + $000D_H$ + ($10_H$ × m),
RSCFDnCFDCmERFLHL: <RSCANn_base> + $000E_H$ + ($10_H$ × m),
RSCFDnCFDCmERFLHH: <RSCANn_base> + $000F_H$ + ($10_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | CRCREG[14:0] | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | ADERR | B0ERR | B1ERR | CERR | AERR | FERR | SERR | ALF | BLF | OVLF | BORF | BOEF | EPF | EWF | BEF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.99 RSCFDnCFDCmERFL Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 30 to 16 | CRCREG[14:0] | CRC Calculation Data (CRC length:15 bits)<br>A CRC value calculated based on the transmit message or receive message is indicated. |
| 15 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 14 | ADERR | ACK Delimiter Error Flag<br>0: No ACK delimiter error is detected.<br>1: ACK delimiter error is detected. |
| 13 | B0ERR | Dominant Bit Error Flag<br>0: No dominant bit error is detected.<br>1: Dominant bit error is detected. |
| 12 | B1ERR | Recessive Bit Error Flag<br>0: No recessive bit error is detected.<br>1: Recessive bit error is detected. |
| 11 | CERR | CRC Error Flag<br>0: No CRC error is detected.<br>1: CRC error is detected. |
| 10 | AERR | ACK Error Flag<br>0: No ACK error is detected.<br>1: ACK error is detected. |
| 9 | FERR | Form Error Flag<br>0: No form error is detected.<br>1: Form error is detected. |

Table 21.99 RSCFDnCFDCmERFL Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 8 | SERR | Stuff Error Flag<br>0: No stuff error is detected.<br>1: Stuff error is detected. |
| 7 | ALF | Arbitration-lost Flag<br>0: No arbitration-lost is detected.<br>1: Arbitration-lost is detected. |
| 6 | BLF | Bus Lock Flag<br>0: No channel bus lock is detected.<br>1: Channel bus lock is detected. |
| 5 | OVLF | Overload Flag<br>0: No overload is detected.<br>1: Overload is detected. |
| 4 | BORF | Bus Off Recovery Flag<br>0: No bus off recovery is detected.<br>1: Bus off recovery is detected. |
| 3 | BOEF | Bus Off Entry Flag<br>0: No bus off entry is detected.<br>1: Bus off entry is detected. |
| 2 | EPF | Error Passive Flag<br>0: No error passive is detected.<br>1: Error passive is detected. |
| 1 | EWF | Error Warning Flag<br>0: No error warning is detected.<br>1: Error warning is detected. |
| 0 | BEF | Bus Error Flag<br>0: No channel bus error is detected.<br>1: Channel bus error is detected. |

See the CAN specification (ISO11898-1) for a description of error occurrence conditions. To clear each flag of this register, the program must write a 0 to the corresponding bit. These flags cannot be set to 1 by the program. If any of these flags is set to 1 at the same time that the program writes 0 to the flag, the flag is still set to 1. The channel reset mode transition clears all of these flags to 0.

If the ERRD bit in the RSCFDnCFDCmCTR register is set to 0 (ie, only the flags for the first error event are displayed) and an error related to bits 14 to 8 of RSCFDnCFDCmERFL is detected, the flag bits are only set to 1 by the error event if bits 14 to 8 were all 0 at the when time the error occurred.

### CRCREG[14:0] Flag

When the CTME bit in the RSCFDnCFDCmCTR register is set to 1 (communication test mode is enabled), if transmit or receive message is a classical CAN frame (CRC length = 15 bits), this flag is updated and the CRC value calculated based on the message can be read. When a CAN FD frame is transmitted or received, the value of CRCREG[20:0] bits in the RSCFDnCFDCmFDCRC register is updated.

### ADERR Flag

This flag is set to 1 when a form error has been detected in the ACK delimiter during transmission.

### B0ERR Flag

This flag is set to 1 when a recessive bit has been detected though a dominant bit was transmitted.

#### B1ERR Flag

This flag is set to 1 when a dominant bit has been detected though a recessive bit was transmitted.

#### CERR Flag

This flag is set to 1 when a CRC error has been detected.

#### AERR Flag

This flag is set to 1 when an ACK error has been detected.

#### FERR Flag

This flag is set to 1 when a form error has been detected.

#### SERR Flag

This flag is set to 1 when a stuff error has been detected.

#### ALF Flag

This flag is set to 1 when an arbitration-lost has been detected.

#### BLF Flag

This flag is set to 1 when 32 consecutive dominant bits have been detected on the CAN bus in channel communication mode. After that, detection of a bus lock is restarted when either of the following conditions is met.

- A recessive bit is detected after the BLF bit has been cleared from 1 to 0.
- The CAN module transitions to channel reset mode and returns to channel communication mode after the BLF bit has been cleared from 1 to 0.

#### OVLF Flag

This flag is set to 1 when the overload frame transmit condition has been detected when performing reception or transmission.

#### BORF Flag

This flag is set to 1 when 11 consecutive recessive bits have been detected 128 times and the CAN module returns from the bus off state. However, this flag is not set to 1 if the CAN module returns from the bus off state in any of the following ways before 11 consecutive recessive bits are detected 128 times.

- The CHMDC[1:0] bits in the RSCFDnCFDCmCTR register are set to $01_B$ (channel reset mode).
- The RTBO bit in the RSCFDnCFDCmCTR register is set to 1 (forcible return from the bus off state is made).
- The BOM[1:0] bits in the RSCFDnCFDCmCTR register are set to $01_B$ (transition to channel halt mode at bus off entry).
- The CHMDC[1:0] bits in the RSCFDnCFDCmCTR register are set to $10_B$ (channel halt mode) before 11 consecutive recessive bits are detected 128 times with the BOM[1:0] bits set to $11_B$ (transition to channel halt mode upon a request from the program during bus off).

### BOEF Flag

This flag is set to 1 when the bus off state is reached (TEC[7:0] value > 255). This flag is also set to 1 if the bus off state is reached when the BOM[1:0] bits in the RSCFDnCFDCmCTR register (m = 0 to 5) are set to $01_B$ (transition to channel halt mode at bus off entry).

### EPF Flag

This flag becomes 1 when the error passive state is reached (REC[7:0] or TEC[7:0] value > 127). This flag becomes 1 only when the REC[7:0] or TEC[7:0] value first exceeds 127. Therefore, if the program writes 0 to this flag while the value of REC[7:0] or TEC[7:0] remains over 127, this bit is not set to 1 until both REC [7:0] and TEC[7:0] values become 127 or less and then the REC[7:0] or TEC[7:0] value exceeds 127 again.

### EWF Flag

This flag is set to 1 only when the REC[7:0] or TEC[7:0] value first exceeds 95. Therefore, if the program writes 0 to this flag while the value of REC[7:0] or TEC[7:0] remains over 95, this bit is not set to 1 until both REC [7:0] and TEC[7:0] values become 95 or less and then the REC[7:0] or TEC[7:0] value exceeds 95 again.

### BEF Flag

This flag is set to 1 when any one of the ADERR, B0ERR, B1ERR, CERR, AERR, FERR, and SERR flags in the RSCFDnCFDCmERFL register is set to 1.

**NOTE**

To clear the flag of this register to 0, use a store instruction to write “0” to the given flag and “1” to other flags.

### 21.5.3.5 RSCFDnCFDCmDCFG — Channel Data Bit Rate Configuration register (m = 0 to 5)

**Access:** RSCFDnCFDCmDCFG register can be read or written in 32-bit units
RSCFDnCFDCmDCFGL, RSCFDnCFDCmDCFGH registers can be read or written in 16-bit units
RSCFDnCFDCmDCFGLL, RSCFDnCFDCmDCFGHL, RSCFDnCFDCmDCFGHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDCmDCFG: <RSCANn_base> + $0500_H$ + ($20_H$ × m)

RSCFDnCFDCmDCFGL: <RSCANn_base> + $0500_H$ + ($20_H$ × m),
RSCFDnCFDCmDCFGH: <RSCANn_base> + $0502_H$ + ($20_H$ × m)

RSCFDnCFDCmDCFGLL: <RSCANn_base> + $0500_H$ + ($20_H$ × m),
RSCFDnCFDCmDCFGHL: <RSCANn_base> + $0502_H$ + ($20_H$ × m),
RSCFDnCFDCmDCFGHH: <RSCANn_base> + $0503_H$ + ($20_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | DSJW[2:0] | | | — | DTSEG2[2:0] | | | DTSEG1[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | DBRP[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.100 RSCFDnCFDCmDCFG Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 26 to 24 | DSJW[2:0] | Data Bit Rate Resynchronization Jump Width Control<br>b26 b25 b24<br>0 0 0: 1 Tq<br>0 0 1: 2 Tq<br>0 1 0: 3 Tq<br>0 1 1: 4 Tq<br>1 0 0: 5 Tq<br>1 0 1: 6 Tq<br>1 1 0: 7 Tq<br>1 1 1: 8 Tq |
| 23 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 22 to 20 | DTSEG2[2:0] | Data Bit Rate Time Segment 2 Control<br>b22 b21 b20<br>0 0 0: Setting prohibited<br>0 0 1: 2 Tq<br>0 1 0: 3 Tq<br>0 1 1: 4 Tq<br>1 0 0: 5 Tq<br>1 0 1: 6 Tq<br>1 1 0: 7 Tq<br>1 1 1: 8 Tq |

**Table 21.100 RSCFDnCFDCmDCFG Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 19 to 16 | DTSEG1[3:0] | Data Bit Rate Time Segment 1 Control<br>b19 b18 b17 b16<br>0 0 0 0: Setting prohibited<br>0 0 0 1: 2 Tq<br>0 0 1 0: 3 Tq<br>0 0 1 1: 4 Tq<br>0 1 0 0: 5 Tq<br>0 1 0 1: 6 Tq<br>0 1 1 0: 7 Tq<br>0 1 1 1: 8 Tq<br>1 0 0 0: 9 Tq<br>1 0 0 1: 10 Tq<br>1 0 1 0: 11 Tq<br>1 0 1 1: 12 Tq<br>1 1 0 0: 13 Tq<br>1 1 0 1: 14 Tq<br>1 1 1 0: 15 Tq<br>1 1 1 1: 16 Tq |
| 15 to 8 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 7 to 0 | DBRP[7:0] | Data Bit Rate Prescaler Division Ratio Setting<br>When the set value = P (0 to 255), the data bit rate prescaler divides fCAN by (P + 1). |

Modify the RSCFDnCFDCmDCFG register in channel reset mode or channel halt mode. Set this register in channel reset mode, and then transition to channel communication mode or channel halt mode. When only classical CAN frames are used in CAN FD mode, set the RSCFDnCFDCmDCFG register to the value equal to the set RSCFDnCFDCmNCFG register value. For the description and settings of bit timing parameters, see **Section 21.11.1, Initial Settings**.

### DSJW[2:0] Bits

These bits are used to specify the resynchronization jump width of data bit rate as a Tq value. A value of 1 to 8 Tq can be set. Specify a value equal to or smaller than the DTSEG2[2:0] bits value.

### DTSEG2[2:0] Bits

These bits are used to specify a Tq value for the length of phase segment 2 (PHASE_SEG2) of the data bit rate.

Allowed values are 2 Tq to 8 Tq, inclusive.

Set a value smaller than the value of the DTSEG1[3:0] bits.

### DTSEG1[3:0] Bits

These bits are used to specify the total length of propagation segment (PROP_SEG) and phase segment 1 (PHASE_SEG1) of data bit rate as a Tq value.

A value from 2 to 16 Tq can be set.

**DBRP[7:0] Bits**

The clock obtained by dividing the CAN clock (fCAN) by the data bit rate prescaler ((DBRP[7:0]) + 1) becomes CANmTq(D) clock (fCANTQ(D)m). One clock of the CANmTq(D) clock becomes one Time Quantum (Tq).

The NBRP[9:0] bit value in RSCFDnCFDCmNCFG register and the DBRP[7:0] bit value should be equal and the two bit rate values be different according to the respective segment values.

When the TDCE bit is set to 1 (Transmitter delay compensation is enabled) in the RSCFDnCFDCmFDCFG register, set the equal value of 1 or less to the bits NBRP[9:0] and DBRP[7:0].

### 21.5.3.6 RSCFDnCFDCmFDCFG — Channel CAN FD Configuration Register (m = 0 to 5)

**Access:** RSCFDnCFDCmFDCFG register can be read or written in 32-bit units
RSCFDnCFDCmFDCFGL, RSCFDnCFDCmFDCFGH registers can be read or written in 16-bit units
RSCFDnCFDCmFDCFGLL, RSCFDnCFDCmFDCFGLH, RSCFDnCFDCmFDCFGHL,
RSCFDnCFDCmFDCFGHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDCmFDCFG: <RSCANn_base> + $0504_H$ + ($20_H$ × m)

RSCFDnCFDCmFDCFGL: <RSCANn_base> + $0504_H$ + ($20_H$ × m),
RSCFDnCFDCmFDCFGH: <RSCANn_base> + $0506_H$ + ($20_H$ × m)

RSCFDnCFDCmFDCFGLL: <RSCANn_base> + $0504_H$ + ($20_H$ × m),
RSCFDnCFDCmFDCFGLH: <RSCANn_base> + $0505_H$ + ($20_H$ × m),
RSCFDnCFDCmFDCFGHL: <RSCANn_base> + $0506_H$ + ($20_H$ × m),
RSCFDnCFDCmFDCFGHH: <RSCANn_base> + $0507_H$ + ($20_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | REFE | FDOE | TMME | GWBRS | GWFDF | GWEN | — | TDCO[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | ESIC | TDCE | TDCOC | — | — | — | — | — | EOCCFG[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

**Table 21.101 RSCFDnCFDCmFDCFG Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 30 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 29 | REFE | Reception Data Edge Filter Enable<br>0: Reception data edge filter disabled<br>1: Reception data edge filter enabled |
| 28 | FDOE | FD-only Mode Enable<br>0: FD-only mode disabled<br>1: FD-only mode enabled |
| 27 | TMME | Transmit Buffer Merge Mode Enable<br>0: Transmit buffer merge mode is disabled.<br>1: Transmit buffer merge mode is enabled. |
| 26 | GWBRS | Gateway BRS<br>0: A frame is transmitted with the BRS bit in the received frame set to 0.<br>1: A frame is transmitted with the BRS bit in the received frame set to 1. |
| 25 | GWFDF | Gateway FDF<br>0: A frame is transmitted regarding the received frame as a classical CAN frame.<br>1: A frame is transmitted regarding the received frame as a CAN FD frame. |
| 24 | GWEN | CAN-CAN FD Gateway Enable<br>0: The CAN-CAN FD gateway is disabled.<br>1: The CAN-CAN FD gateway is enabled. |
| 23 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 22 to 16 | TDCO[6:0] | Transmitter Delay Compensation Offset<br>These bits are set to the transmitter delay compensation offset value. |
| 15 to 11 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

Table 21.101 RSCFDnCFDCmFDCFG Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 10 | ESIC | Error State Display Mode Select<br>0: Always displays the node error state.<br>1: When the node is not in the error passive state: Displays the message buffer error state.<br>When the node is in the error passive state: Displays the node error state. |
| 9 | TDCE | Transmitter Delay Compensation Enable<br>0: Transmitter delay compensation is disabled.<br>1: Transmitter delay compensation is enabled. |
| 8 | TDCOC | Transmitter Delay Compensation Measurement Select<br>0: Measurement and offset<br>1: Only offset |
| 7 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 to 0 | EOCCFG[2:0] | Error Occurrence Counting Method Select<br>b2 b1 b0<br>0 0 0: All transmit messages and receive messages<br>0 0 1: All transmit messages<br>0 1 0: All receive messages<br>0 1 1: Setting prohibited<br>1 0 0: Only data phase of transmitted or received CAN FD message<br>1 0 1: Only data phase of transmitted CAN FD message<br>1 1 0: Only data phase of received CAN FD message<br>1 1 1: Setting prohibited |

### REFE bit

Setting this bit to 1 enables reception data edge filtering when the idle condition is detected, and a dominant level with less than 2 time quanta is ignored. A dominant level with more than or equal to 2 time quanta is detected as an edge.

Modify this bit only in channel reset mode.

### FDOE bit

Setting this bit to 1 enables FD-only mode. When data is transmitted, a CAN FD frame will be sent regardless of the settings to the CFFDF bit in the RSCFDnCFDCFFDCSTSk register or the TMFDF bit in the RSCFDnCFDTMFDCTRp register. When a Classical CAN frame is received, a form error is detected. Modify this bit only in channel reset mode.

### TMME Bit

Setting this bit to 1 enables transmit buffer merge mode. Modify this bit only in channel reset mode or channel halt mode.

### GWBRS Bit

When the GWEN bit is 1, the BRS bit in a CAN FD frame to be transmitted by the gateway function is set. Write 0 to this bit to clear the GWBRS bit to 0. Modify this bit only in channel reset mode.

### GWFDF Bit

When the GWEN bit is 1, the FDF bit in a CAN FD frame to be transmitted by the gateway function is set. Modify this bit only in channel reset mode.

### GWEN Bit

This bit is used to control the operation of the transmit/receive FIFO buffer with the CFM[1:0] bits in the RSCFDnCFDCFCCk register set to $10_B$ (gateway mode).

Setting this bit to 1 enables the CAN-CAN FD gateway, enabling transmission in a format different from that of frames received by the gateway function. Received frames are replaced in accordance with the settings of the GWFDF bit and the GWBRS bit. When the DLC value in the received classical CAN frame is $1001_B$ or more and the GWFDF bit is 1 (CAN FD frame), the DLC value is replaced with $1000_B$.

While this bit is 1, do not perform routing for the following frames by using the gateway function.

- CAN FD frames with a payload length of more than 8 bytes
- Remote frames

While this bit is 1, only the following frame should be transmitted from the corresponding channel by setting of GWFDF.

- When GWFDF bit is set to 0, only classical CAN frame should be transmitted.
- When GWFDF bit is set to 1, only CAN FD frame should be transmitted.

**Table 21.102** shows the settings and formats of transmit frame and receive frame while the CAN-CAN FD gateway is enabled.

**Table 21.102 Operation when the CAN-CAN FD Gateway is Enabled**

| Receive Frame | | | | Transmit Frame | | |
|---|---|---|---|---|---|---|
| Format | BRS Bit | Received DLC Value | GWFDF Bit | Format | BRS Bit | DLC Value to be Transmitted |
| Classical CAN | None | DLC ≤ $1000_B$ | 0 | Classical CAN | None | Not replaced |
| | | DLC > $1000_B$ | | | | |
| CAN FD | Arbitrary | DLC ≤ $1000_B$ | | | | |
| Classical CAN | None | DLC ≤ $1000_B$ | 1 | CAN FD | According to GWBRS bit setting | Not replaced |
| | | DLC >$1000_B$ | | | | Replaced with $1000_B$ |
| CAN FD | Arbitrary | DLC ≤ $1000_B$ | | | | Not replaced |

### TDCO[6:0] Bits

These bits are set to the SSP offset value. How to use this value depends on the TDCOC bit in the RSCFDnCFDCmFDCFG register.

When the TDCOC bit is 0, the transmitter delay compensation result equals to the total value of the measured delay value and the TDCO[6:0] value (rounded to the nearest integer Tq).

When the TDCOC bit is 1, the transmitter delay compensation result equals to the TDCO[6:0] value.

The SSP offset value = (set value of TDCO[6:0] bits + 1).

Modify these bits only in channel reset mode or channel halt mode.

### ESIC Bit

When the ESIC bit is set to 1, if the channel is in the error active state, the ESI bit value (CFESI bit in the RSCFDnCFDRSCFDnCFDCFFDCSTSk register or TMESI bit in the RSCFDnCFDTMFDCTRp register) set in the transmit/receive FIFO buffer or transmit buffer is transmitted as an ESI bit value of the transmit message. When the channel is in the error passive state or the ESIC bit is 0, the channel status is transmitted as an ESI bit value. Modify this bit only in channel reset mode or channel halt mode.

**Table 21.103 ESI Value to Be Transmitted**

| ESIC Value | Channel Status | ESI Value to be Transmitted |
|---|---|---|
| 0 | Error active | 0 (error active node) |
| | Error passive | 1 (error passive node) |
| 1 | Error active | ESI value set in the transmit/receive FIFO buffer or transmit buffer (CFESI bit in the RSCFDnCFDCFFDCSTSk register or TMESI bit in the RSCFDnCFDTMFDCTRp register) |
| | Error passive | 1 (error passive node) |

### TDCE Bit

Setting this bit to 1 enables transmitter delay compensation. Modify this bit only in channel reset mode or channel halt mode.

### TDCOC Bit

When this bit is 0, the SSP position is defined by the total of the measured delay value and the SSP offset value (fixed value).

When this bit is 1, the SSP position is defined only by the SSP offset value.

Modify this bit only in channel reset mode or channel halt mode.

### EOCCFG[2:0] Bits

These bits are used to select a frame format and a transmission/reception direction when the error occurrence counter counts CAN bus errors.

Modify these bits only in channel reset mode or channel halt mode.

### 21.5.3.7 RSCFDnCFDCmFDCTR — Channel CAN FD Control Register (m = 0 to 5)

**Access:** RSCFDnCFDCmFDCTR register can be read or written in 32-bit units
RSCFDnCFDCmFDCTRL register can be read or written in 16-bit units
RSCFDnCFDCmFDCTRLL register can be read or written in 8-bit units

**Address:** RSCFDnCFDCmFDCTR: <RSCANn_base> + $0508_H$ + ($20_H$ × m)

RSCFDnCFDCmFDCTRL: <RSCANn_base> + $0508_H$ + ($20_H$ × m)

RSCFDnCFDCmFDCTRLL: <RSCANn_base> + $0508_H$ + ($20_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SOCCLR | EOCCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 21.104 RSCFDnCFDCmFDCTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | SOCCLR | Successful Communication Occurrence Counter Clear<br>Setting the SOCCLR bit to 1 clears the successful communication occurrence counter. This bit is always read as 0. |
| 0 | EOCCLR | Error Occurrence Counter Clear<br>Setting the EOCCLR bit to 1 clears the error occurrence counter. This bit is always read as 0. |

#### SOCCLR Bit

Setting this bit to 1 clears the successful communication occurrence counter (SOC[7:0] bits in the RSCFDnCFDCmFDSTS register). This bit is automatically cleared to 0.

#### EOCCLR Bit

Setting this bit to 1 clears the error occurrence counter (EOC[7:0] bits in the RSCFDnCFDCmFDSTS register). This bit is automatically cleared to 0.

### 21.5.3.8 RSCFDnCFDCmFDSTS — Channel CAN FD Status Register (m = 0 to 5)

**Access:** RSCFDnCFDCmFDSTS register can be read or written in 32-bit units
RSCFDnCFDCmFDSTSL register can be read or written in 16-bit units
RSCFDnCFDCmFDSTSH register is a read-only register that can be read in 16-bit units
RSCFDnCFDCmFDSTSLL, RSCFDnCFDCmFDSTSLH registers can be read or written in 8-bit units
RSCFDnCFDCmFDSTSHL, RSCFDnCFDCmFDSTSHH registers are the read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDCmFDSTS: <RSCANn_base> + $050C_H$ + ($20_H$ × m)

RSCFDnCFDCmFDSTSL: <RSCANn_base> + $050C_H$ + ($20_H$ × m),
RSCFDnCFDCmFDSTSH: <RSCANn_base> + $050E_H$ + ($20_H$ × m)

RSCFDnCFDCmFDSTSLL: <RSCANn_base> + $050C_H$ + ($20_H$ × m),
RSCFDnCFDCmFDSTSLH: <RSCANn_base> + $050D_H$ + ($20_H$ × m),
RSCFDnCFDCmFDSTSHL: <RSCANn_base> + $050E_H$ + ($20_H$ × m),
RSCFDnCFDCmFDSTSHH: <RSCANn_base> + $050F_H$ + ($20_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SOC[7:0] | | | | | | | | EOC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | SOCO | EOCO | TDCVF | TDCR[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W*1 | R/W*1 | R/W*1 | R | R | R | R | R | R | R |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.105 RSCFDnCFDCmFDSTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | SOC[7:0] | Successful Communication Occurrence Counter<br>The successful communication occurrence counter value can be read. |
| 23 to 16 | EOC[7:0] | Error Occurrence Counter<br>The error occurrence counter value can be read. |
| 15 to 10 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 9 | SOCO | Successful Communication Occurrence Counter Overflow Flag<br>0: The successful communication occurrence counter does not overflow.<br>1: The successful communication occurrence counter has overflowed. |
| 8 | EOCO | Error Occurrence Counter Overflow Flag<br>0: The error occurrence counter does not overflow.<br>1: The error occurrence counter has overflowed. |
| 7 | TDCVF | Transmitter Delay Compensation Violation Flag<br>0: No transmitter delay compensation violation is present.<br>1: A transmitter delay compensation violation is present. |
| 6 to 0 | TDCR[6:0] | Transmitter Delay Compensation Result Status<br>The transmitter delay compensation result can be read. |

#### SOC[7:0] Bits

These bits show the successful communication occurrence counter value. The successful communication occurrence counter is incremented upon completion of message reception or transmission without an error. This counter stops counting when it reaches $FF_H$. In loopback mode, this counter is incremented twice.

These bits are cleared to 0 by writing 1 to the SOCCLR bit in the RSCFDnCFDCmFDCTR register. These bits are 0 in channel reset mode.

#### EOC[7:0] Bits

These bits show the error occurrence counter value. The error occurrence counter is incremented each time an error occurs according to the condition specified by the EOCCFG[2:0] bits in the RSCFDnCFDCmFDCFG register. This counter stops counting when it reaches $FF_H$.

These bits are cleared to 0 by writing 1 to the EOCCLR bit in the RSCFDnCFDCmFDCTR register. These bits are 0 in channel reset mode.

#### SOCO Flag

This bit indicates that successful communication occurrence counter overflow has occurred.

This flag is set to 1 when message reception or transmission is completed while the SOC[7:0] value has reached $FF_H$. This flag is 0 in channel reset mode.

#### EOCO Flag

This bit indicates that error occurrence counter overflow has occurred.

This flag is set to 1 when a CAN bus error is detected under the condition specified by the EOCCFG[2:0] bits in the RSCFDnCFDCmFDCFG register when the EOC[7:0] value has reached $FF_H$. This flag is 0 in channel reset mode.

#### TDCVF Flag

This bit indicates violation of transmitter delay compensation.

The transmit data is compared with the reception CAN bus level delayed due to the transceiver's loop delay. This delay changes due to physical factors such as temperature. Because the TDCR[6:0] flags are updated for each message, temporary maximum delay cannot be confirmed.

This bit is set to 1 when the transmitter delay compensation exceeds the maximum compensation 3 CANm bit times - 2 fCAN (CANm bit time is the value of data bit rate).

This flag is 0 in channel reset mode.

#### TDCR[6:0] Flags

These bits indicate the transmitter delay compensation result as a multiple of CAN clock frequency (fCAN).

This result depends on the settings of the TDCOC bit and TDCO[6:0] bits in the RSCFDnCFDCmFDCFG register.

This flag is updated at a falling edge between the FDF bit and res bit when the TDCE bit (transmitter delay compensation enable) in the RSCFDnCFDCmFDCFG register is set to 1 and also the TDCOC bit (transmitter delay compensation measurement select) in the RSCFDnCFDCmFDCFG register is set to 0.

This flag is 0 in channel reset mode.

### 21.5.3.9 RSCFDnCFDCmFDCRC — Channel CAN FD CRC Register (m = 0 to 5)

**Access:** RSCFDnCFDCmFDCRC register is a read-only register that can be read in 32-bit units
RSCFDnCFDCmFDCRCL, RSCFDnCFDCmFDCRCH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDCmFDCRCLL, RSCFDnCFDCmFDCRCLH, RSCFDnCFDCmFDCRCHL, RSCFDnCFDCmFDCRCHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDCmFDCRC: <RSCANn_base> + $0510_H$ + ($20_H$ × m)

RSCFDnCFDCmFDCRCL: <RSCANn_base> + $0510_H$ + ($20_H$ × m),
RSCFDnCFDCmFDCRCH: <RSCANn_base> + $0512_H$ + ($20_H$ × m)

RSCFDnCFDCmFDCRCLL: <RSCANn_base> + $0510_H$ + ($20_H$ × m),
RSCFDnCFDCmFDCRCLH: <RSCANn_base> + $0511_H$ + ($20_H$ × m),
RSCFDnCFDCmFDCRCHL: <RSCANn_base> + $0512_H$ + ($20_H$ × m),
RSCFDnCFDCmFDCRCHH: <RSCANn_base> + $0513_H$ + ($20_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | SCNT[3:0] | | | | — | — | — | CRCREG[20:16] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CRCREG[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.106 RSCFDnCFDCmFDCRC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | Reserved | When read, the value after reset is returned. |
| 27 to 24 | SCNT[3:0] | Stuff count bit<br>Indicate a value of the stuff count in a CAN FD frame.<br>Bits 27 to 25 indicate the Gray-coded value of the stuff bit count module 8 in the transmitted/received frames.<br>Bit 24 indicates an even parity value of bits 27 to 25. |
| 23 to 21 | Reserved | When read, the value after reset is returned. |
| 20 to 0 | CRCREG[20:0] | CRC Calculation Data (CRC Length:17 Bit or 21 Bit)<br>These bits show the CRC value calculated based on the transmit message or receive message.<br>When the CRC length is 17 bits, bits b20 to b17 are read as 0. |

#### SCNT[3:0] Flags

When the CTME bit in the RSCFDnCFDCmCTR register is set to 1 (communication test mode enabled), a stuff count bit value of the CAN FD frame can be read if a message transmitted/received is a CAN FD frame. When the CTME bit is 0 (communication test mode disabled), this flag is always read as 0. This flag is updated at the first bit in the CRC field of the CAN FD frame. These bits are cleared to 0 in channel reset mode.

**CRCREG[20:0] Flag**

When the CTME bit in the RSCFDnCFDCmCTR register is 1 (communication test mode enabled), if transmit or receive message is a CAN FD frame (CRC length = 17 or 21 bits), these flags are updated and the CRC value calculated based on the message can be read. When the CRC length of the message is 17 bits, bits b20 to b17 are always read as 0. When a classical CAN frame is transmitted or received, the CRCREG[14:0] value in the RSCFDnCFDCmERFL register is updated. When the CTME bit is 0 (communication test mode disabled), these bits are always read as 0.

## 21.5.4 Details of Global-Related Registers

### 21.5.4.1 RSCFDnCFDGCFG — Global Configuration Register

**Access:** RSCFDnCFDGCFG register can be read or written in 32-bit units
RSCFDnCFDGCFGL, RSCFDnCFDGCFGH registers can be read or written in 16-bit units
RSCFDnCFDGCFGLL, RSCFDnCFDGCFGLH, RSCFDnCFDGCFGHL, RSCFDnCFDGCFGHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDGCFG: <RSCANn_base> + $0084_H$

RSCFDnCFDGCFGL: <RSCANn_base> + $0084_H$,
RSCFDnCFDGCFGH: <RSCANn_base> + $0086_H$

RSCFDnCFDGCFGLL: <RSCANn_base> + $0084_H$,
RSCFDnCFDGCFGLH: <RSCANn_base> + $0085_H$,
RSCFDnCFDGCFGHL: <RSCANn_base> + $0086_H$,
RSCFDnCFDGCFGHH: <RSCANn_base> + $0087_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ITRCP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSBTCS[2:0] | | | TSSS | TSP[3:0] | | | | — | — | CMPOC | DCS | MME | DRE | DCE | TPRI |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.107 RSCFDnCFDGCFG Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | ITRCP[15:0] | Interval Timer Prescaler Set<br>When these bits are set to M, the pclk is divided by M.<br>Setting $0000_H$ is prohibited when the interval timer is in use. |
| 15 to 13 | TSBTCS[2:0] | Timestamp Clock Source Select<br>b15 b14 b13<br>0 0 0: Channel 0 nominal bit time clock<br>0 0 1: Channel 1 nominal bit time clock<br>0 1 0: Channel 2 nominal bit time clock<br>0 1 1: Channel 3 nominal bit time clock<br>1 0 0: Channel 4 nominal bit time clock<br>1 0 1: Channel 5 nominal bit time clock<br>1 1 0: Setting prohibited<br>1 1 1: Setting prohibited |
| 12 | TSSS | Timestamp Source Select<br>0: pclk/2*1<br>1: Nominal bit time clock |

**Table 21.107 RSCFDnCFDGCFG Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 to 8 | TSP[3:0] | Timestamp Clock Source Division<br>b11 b10 b9 b8<br>0 0 0 0: Not divided<br>0 0 0 1: Divided by 2<br>0 0 1 0: Divided by 4<br>0 0 1 1: Divided by 8<br>0 1 0 0: Divided by 16<br>0 1 0 1: Divided by 32<br>0 1 1 0: Divided by 64<br>0 1 1 1: Divided by 128<br>1 0 0 0: Divided by 256<br>1 0 0 1: Divided by 512<br>1 0 1 0: Divided by 1024<br>1 0 1 1: Divided by 2048<br>1 1 0 0: Divided by 4096<br>1 1 0 1: Divided by 8192<br>1 1 1 0: Divided by 16384<br>1 1 1 1: Divided by 32768 |
| 7, 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 | CMPOC | Payload Overflow Mode Select<br>0: No message is stored.<br>1: Messages are stored and payloads exceeding the buffer size are discarded. |
| 4 | DCS | CAN Clock Source Select*2<br>0: clkc<br>1: clk_xincan |
| 3 | MME | Mirror Function Enable<br>0: Mirror function is disabled.<br>1: Mirror function is enabled. |
| 2 | DRE | DLC Replacement Enable<br>0: DLC replacement is disabled.<br>1: DLC replacement is enabled. |
| 1 | DCE | DLC Check Enable<br>0: DLC check is disabled.<br>1: DLC check is enabled. |
| 0 | TPRI | Transmit Priority Select<br>0: ID priority<br>1: Transmit buffer number priority |

Note 1. When specifying pclk/2 as the timestamp counter count source, set bits TSBTCS[2:0] to $000_B$.

Note 2. For the CAN clock frequency settings, see **Table 21.8, Range of Operating Frequency Depending on the Transfer Rate and the Number of Channels in Use in the RH850/F1K**.

Modify the RSCFDnCFDGCFG register only in global reset mode.

### ITRCP[15:0] Bits

These bits are used to set a clock source division value of the interval timer for FIFO buffers. For details, see **Section 21.9.3.1, Interval Transmission Function**.

### TSBTCS[2:0] Bits

When the TSSS bit is 1, these bits are used to select the channel of the nominal bit time clock that will be the clock source of the timestamp counter. However, do not select the channel that handles the CAN FD frames.

#### TSSS Bit

This bit is used to select a clock source of the timestamp counter. Select pclk if there is no channel that handles only classical CAN frames.

#### TSP[3:0] Bits

A clock obtained by dividing the clock source selected with the TSBTCS[2:0] bits and TSSS bit by the TSP[3:0] bits is used as the timestamp counter count source.

#### CMPOC Bit

This bit is used to select operation in case the payload length of received message exceeds the payload storage size of the storage buffer.

When this bit is 0, the received message in which the payload overflows is not stored in the buffer.

When this bit is 1, the received message in which the payload overflows is stored in the buffer. At this time, payloads exceeding the buffer's payload storage size are discarded.

The buffer’s payload storage size is set by the following bits.

- Receive buffer: RMPLS[1:0] bits in the RSCFDnCFDRMNB register
- Receive FIFO buffer: RFPLS[2:0] bits in the RSCFDnCFDRFCCx register
- Transmit/receive FIFO buffer: CFPLS[2:0] bits in the RSCFDnCFDCFCCk register

#### DCS Bit

When this bit is set to 0, clkc is used as the clock source of the CAN clock (fCAN).

When this bit is set to 1, clk_xincan is used as the clock source of the CAN clock (fCAN).

For the CAN clock frequency settings, see **Table 21.8, Range of Operating Frequency Depending on the Transfer Rate and the Number of Channels in Use in the RH850/F1K**.

#### MME Bit

Setting this bit to 1 makes the mirror function available.

#### DRE Bit

When the DRE bit is set to 1, the DLC value of the receive rule is stored in the buffer instead of the DLC value of the received message after the DLC value has passed through the DLC filter. In this case, a value of $00_H$ is stored in each data byte beyond the DLC value of the receive rule.

The DLC replacement function is only available when the DCE bit is set to 1 (DLC check is enabled).

#### DCE Bit

Setting this bit to 1 makes the DLC check function available. When disabling the DLC check function, set the GAFLDLC[3:0] bits in the RSCFDnCFDGAFLP0_j register to $0000_B$ before clearing the DCE bit in the RSCFDnCFDGCFG register to 0.

**TPRI Bit**

This bit is used to set the transmit priority.

When this bit is set to 0, ID priority is selected and the transmit priority complies with the CAN bus arbitration rule (ISO11898-1 specifications). When this bit is set to 1, transmit buffer number priority is selected and the lowest transmit buffer number of those has the highest priority.

While the transmit queue is in use, this bit should be set to 0.

### 21.5.4.2 RSCFDnCFDGCTR — Global Control Register

**Access:** RSCFDnCFDGCTR register can be read or written in 32-bit units
RSCFDnCFDGCTRL, RSCFDnCFDGCTRH registers can be read or written in 16-bit units
RSCFDnCFDGCTRLL, RSCFDnCFDGCTRLH, RSCFDnCFDGCTRHL registers can be read or written in 8-bit units

**Address:** RSCFDnCFDGCTR: <RSCANn_base> + $0088_H$

RSCFDnCFDGCTRL: <RSCANn_base> + $0088_H$,
RSCFDnCFDGCTRH: <RSCANn_base> + $008A_H$

RSCFDnCFDGCTRLL: <RSCANn_base> + $0088_H$,
RSCFDnCFDGCTRLH: <RSCANn_base> + $0089_H$,
RSCFDnCFDGCTRHL: <RSCANn_base> + $008A_H$

**Value after reset:** $0000\ 0005_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TSRST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | CMPOFIE | THLEIE | MEIE | DEIE | — | — | — | — | — | GSLPR | GMDC[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

**Table 21.108 RSCFDnCFDGCTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 17 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 16 | TSRST | Timestamp Counter Reset<br>Setting the TSRST bit to 1 resets the timestamp counter. This bit is always read as 0. |
| 15 to 12 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 11 | CMPOFIE | Payload Overflow Interrupt Enable<br>0: A payload overflow interrupt is disabled.<br>1: A payload overflow interrupt is enabled. |
| 10 | THLEIE | Transmit History Buffer Overflow Interrupt Enable<br>0: Transmit history buffer overflow interrupt is disabled.<br>1: Transmit history buffer overflow interrupt is enabled. |
| 9 | MEIE | FIFO Message Lost Interrupt Enable<br>0: FIFO message lost interrupt is disabled.<br>1: FIFO message lost interrupt is enabled. |
| 8 | DEIE | DLC Error Interrupt Enable<br>0: DLC error interrupt is disabled.<br>1: DLC error interrupt is enabled. |
| 7 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 | GSLPR | Global Stop Mode<br>0: Other than global stop mode<br>1: Global stop mode |
| 1, 0 | GMDC[1:0] | Global Mode Select<br>b1 b0<br>0 0: Global operating mode<br>0 1: Global reset mode<br>1 0: Global test mode<br>1 1: Setting prohibited |

#### TSRST Bit

This bit is used to reset the timestamp counter. When this bit is set to 1, the RSCFDnCFDGTSC register is cleared to $0000_H$.

#### CMPOFIE Bit

When the CMPOF flag in the RSCFDnCFDGERFL register is set to 1 after the CMPOFIE bit is set to 1, an interrupt request is generated. Modify this bit only in global reset mode.

#### THLEIE Bit

When the THLEIE bit is set to 1 and the THLES flag in the RSCFDnCFDGERFL register is set to 1, an interrupt request is generated. Modify this bit only in global reset mode.

#### MEIE Bit

When the MEIE bit is set to 1 and the MES flag in the RSCFDnCFDGERFL register is set to 1, an interrupt request is generated. Modify this bit only in global reset mode.

#### DEIE Bit

When the DEIE bit is set to 1 and the DEF flag in the RSCFDnCFDGERFL register is set to 1, an interrupt request is generated. Modify this bit only in global reset mode.

#### GSLPR Bit

Setting this bit to 1 places the RS-CANFD module into global stop mode.
Clearing this bit to 0 makes the RS-CANFD module leave from global stop mode.
This bit should not be modified in global operating mode or global test mode.

#### GMDC[1:0] Bits

These bits are used to select the mode of entire RS-CANFD module (global operating mode, global reset mode, or global test mode). For details, see **Section 21.7.1, Global Modes**. Setting the GSLPR bit to 1 when in global reset mode places the RS-CANFD module into global stop mode.

### 21.5.4.3 RSCFDnCFDGSTS — Global Status Register

**Access:** RSCFDnCFDGSTS register is a read-only register that can be read in 32-bit units
RSCFDnCFDGSTSL register is a read-only register that can be read in 16-bit units
RSCFDnCFDGSTSLL register is a read-only register that can be read in 8-bit units

**Address:** RSCFDnCFDGSTS: <RSCANn_base> + $008C_H$

RSCFDnCFDGSTSL: <RSCANn_base> + $008C_H$

RSCFDnCFDGSTSLL: <RSCANn_base> + $008C_H$

**Value after reset:** $0000\ 000D_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | GRAMINIT | GSLPSTS | GHLTSTS | GRSTSTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.109 RSCFDnCFDGSTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is returned. |
| 3 | GRAMINIT | CAN RAM Initialization Status Flag<br>0: CAN RAM initialization is completed.<br>1: CAN RAM initialization is ongoing. |
| 2 | GSLPSTS | Global Stop Status Flag<br>0: Not in global stop mode<br>1: In global stop mode |
| 1 | GHLTSTS | Global Test Status Flag<br>0: Not in global test mode<br>1: In global test mode |
| 0 | GRSTSTS | Global Reset Status Flag<br>0: Not in global reset mode<br>1: In global reset mode |

#### GRAMINIT Flag

This flag indicates the initialization status of the CAN RAM.

This flag is set to 1 after the MCU has been reset, and is cleared to 0 when CAN RAM initialization is completed.

#### GSLPSTS Flag

This flag is set to 1 when the CAN module has transitioned to global stop mode, and is cleared to 0 when the CAN module has returned from global stop mode.

#### GHLTSTS Flag

This flag is set to 1 when the CAN module has transitioned to global test mode, and is cleared to 0 when the CAN module has exited global test mode.

**GRSTSTS Flag**

This flag is set to 1 when the CAN module has transitioned to global reset mode, and is cleared to 0 when the CAN module has exited global reset mode. This flag remains 1 even when the CAN module has transitioned from global reset mode to global stop mode.

### 21.5.4.4 RSCFDnCFDGERFL — Global Error Flag Register

**Access:** RSCFDnCFDGERFL register can be read or written in 32-bit units
RSCFDnCFDGERFLL, RSCFDnCFDGERFLH registers can be read or written in 16-bit units
RSCFDnCFDGERFLLL, RSCFDnCFDGERFLHL registers can be read or written in 8-bit units

**Address:** RSCFDnCFDGERFL: <RSCANn_base> + $0090_H$

RSCFDnCFDGERFLL: <RSCANn_base> + $0090_H$,
RSCFDnCFDGERFLH: <RSCANn_base> + $0092_H$

RSCFDnCFDGERFLLL: <RSCANn_base> + $0090_H$,
RSCFDnCFDGERFLHL: <RSCANn_base> + $0092_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | EEF5 | EEF4 | EEF3 | EEF2 | EEF1 | EEF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | CMPOF | THLES | MES | DEF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W*1 | R | R | R/W*1 |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.110 RSCFDnCFDGERFL Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 22 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 21 | EEF5 | ECC Error Flag for Channel 5<br>0: No 2-bit ECC error when deciding transmission priority<br>1: A 2-bit ECC error when deciding transmission priority |
| 20 | EEF4 | ECC Error Flag for Channel 4<br>0: No 2-bit ECC error when deciding transmission priority<br>1: A 2-bit ECC error when deciding transmission priority |
| 19 | EEF3 | ECC Error Flag for Channel 3<br>0: No 2-bit ECC error when deciding transmission priority<br>1: A 2-bit ECC error when deciding transmission priority |
| 18 | EEF2 | ECC Error Flag for Channel 2<br>0: No 2-bit ECC error when deciding transmission priority<br>1: A 2-bit ECC error when deciding transmission priority |
| 17 | EEF1 | ECC Error Flag for Channel 1<br>0: No 2-bit ECC error when deciding transmission priority<br>1: A 2-bit ECC error when deciding transmission priority |
| 16 | EEF0 | ECC Error Flag for Channel 0<br>0: No 2-bit ECC error when deciding transmission priority<br>1: A 2-bit ECC error when deciding transmission priority |
| 15, 14 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 13 to 8 | Reserved | When read, the undefined value is returned. When writing, write the value after reset. |
| 7, 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 | Reserved | When read, the undefined value is returned. When writing, write the value after reset. |

Table 21.110 RSCFDnCFDGERFL Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 | CMPOF | Payload Overflow Flag<br>0: No payload overflow has occurred.<br>1: A payload overflow has occurred. |
| 2 | THLES | Transmit History Buffer Overflow Status Flag<br>0: No transmit history buffer overflow has occurred.<br>1: A transmit history buffer overflow has occurred. |
| 1 | MES | FIFO Message Lost Status Flag<br>0: No FIFO message lost error has occurred.<br>1: A FIFO message lost error has occurred. |
| 0 | DEF | DLC Error Flag<br>0: No DLC error has occurred.<br>1: A DLC error has occurred. |

All flags in the RSCFDnCFDGERFL register are cleared to 0 in global reset mode.

### EEFm Flag

When a 2-bit ECC error is detected during the transmission priority determination of channel m (m = 0 to 5), the EEFm flag is set to 1, disabling message transmission. This flag can be cleared to 0 by writing 0 by the program.

### CMPOF Flag

When a payload overflow occurs in any of channel m (m = 0 to 5), the CMPOF flag is set to 1. This flag can be cleared to 0 by writing 0 to this bit by the program.

### THLES Flag

The THLES flag is set to 1 when any one of the THLELT flags in the RSCFDnCFDTHLSTSm register (m = 0 to 5) is set to 1.

This flag is cleared to 0 when the THLELT flags of all channels are set to 0.

### MES Flag

The MES flag is set to 1 when any one of the RFMLT flags in the RSCFDnCFDRFSTSx register (x = 0 to 7) or the CFMLT flags in the RSCFDnCFDCFSTSk register (k = 0 to 17) is set to 1.

This flag is cleared to 0 when all RFMLT flags and CFMLT flags are set to 0.

### DEF Flag

The DEF flag is set to 1 when an error has been detected during the DLC check. The program can clear this flag by writing 0 to this bit.

**NOTE**

To clear the flag of this register to 0, use a store instruction to write 0 to the given flag and 1 to the other flags.

### 21.5.4.5 RSCFDnCFDGTSC — Global Timestamp Counter Register

**Access:** RSCFDnCFDGTSC register is a read-only register that can be read in 32-bit units.
RSCFDnCFDGTSCL register is a read-only register that can be read in 16-bit units.

**Address:** RSCFDnCFDGTSC: <RSCANn_base> + $0094_H$

RSCFDnCFDGTSCL: <RSCANn_base> + $0094_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.111 RSCFDnCFDGTSC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. |
| 15 to 0 | TS[15:0] | Timestamp Value<br>The timestamp counter value can be read.<br>Counter Value: $0000_H$ to $FFFF_H$ |

#### TS[15:0] Bits

When the TS[15:0] bits are read, the read value shows the timestamp counter (16-bit free-running counter) value at that time. When the SOF is detected, the TS[15:0] value is captured and later stored in the receive buffer or the FIFO buffer. Furthermore, the TS[15:0] value is stored in the transmit history buffer. The timestamp counter is initialized in global reset mode.

The timestamp counter starts and stops counting differently, depending on the count source.

- When the TSSS bit in the RSCFDnCFDGCFG register is 0 (pclk):
  The timestamp counter starts counting when the RSCAN module has transitioned to global operating mode.
  This counter stops counting when the RSCAN module has transitioned to global stop mode or global test mode.
- When the TSSS bit is 1 (CANm nominal bit time clock):
  The timestamp counter starts counting when the corresponding channel has transitioned to channel communication mode.
  This counter stops counting when the corresponding channel has transitioned to channel reset mode or channel halt mode.

### 21.5.4.6 RSCFDnCFDGTINTSTS0 — Global TX Interrupt Status Register 0

**Access:** RSCFDnCFDGTINTSTS0 register is a read-only register that can be read in 32-bit units
RSCFDnCFDGTINTSTS0L, RSCFDnCFDGTINTSTS0H registers are read-only registers that can be read in 16-bit units
RSCFDnCFDGTINTSTS0LL, RSCFDnCFDGTINTSTS0LH, RSCFDnCFDGTINTSTS0HL, RSCFDnCFDGTINTSTS0HH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDGTINTSTS0: <RSCANn_base> + $0460_H$

RSCFDnCFDGTINTSTS0L: <RSCANn_base> + $0460_H$,
RSCFDnCFDGTINTSTS0H: <RSCANn_base> + $0462_H$

RSCFDnCFDGTINTSTS0LL: <RSCANn_base> + $0460_H$,
RSCFDnCFDGTINTSTS0LH: <RSCANn_base> + $0461_H$,
RSCFDnCFDGTINTSTS0HL: <RSCANn_base> + $0462_H$,
RSCFDnCFDGTINTSTS0HH: <RSCANn_base> + $0463_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | THIF3 | CFTIF3 | TQIF3 | TAIF3 | TSIF3 | — | — | — | THIF2 | CFTIF2 | TQIF2 | TAIF2 | TSIF2 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R*1 | R*1 | R*1 | R*1 | R*1 | R | R | R | R*1 | R*1 | R*1 | R*1 | R*1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | THIF1 | CFTIF1 | TQIF1 | TAIF1 | TSIF1 | — | — | — | THIF0 | CFTIF0 | TQIF0 | TAIF0 | TSIF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R*1 | R*1 | R*1 | R*1 | R*1 | R | R | R | R*1 | R*1 | R*1 | R*1 | R*1 |

Note 1. This bit is automatically cleared in the global reset or channel reset mode.

**Table 21.112 RSCFDnCFDGTINTSTS0 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | When read, the value after reset is returned. |
| 28 | THIF3 | Channel 3 Transmit History Interrupt Status Flag<br>0: Transmit history interrupt is not requested.<br>1: Transmit history interrupt is requested. |
| 27 | CFTIF3 | Channel 3 Transmit/receive FIFO Transmit Interrupt Status Flag<br>0: Transmit/receive FIFO transmit interrupt is not requested.<br>1: Transmit/receive FIFO transmit interrupt is requested. |
| 26 | TQIF3 | Channel 3 Transmit Queue Interrupt Status Flag<br>0: Transmit queue interrupt is not requested.<br>1: Transmit queue interrupt is requested. |
| 25 | TAIF3 | Channel 3 Transmit Buffer Abort Interrupt Status Flag<br>0: Transmit buffer abort interrupt is not requested.<br>1: Transmit buffer abort interrupt is requested |
| 24 | TSIF3 | Channel 3 Transmit Buffer Transmit Complete Interrupt Status Flag<br>0: Transmit buffer transmit complete interrupt is not requested.<br>1: Transmit buffer transmit complete interrupt is requested. |
| 23 to 21 | Reserved | When read, the value after reset is returned. |
| 20 | THIF2 | Channel 2 Transmit History Interrupt Status Flag<br>0: Transmit history interrupt is not requested.<br>1: Transmit history interrupt is requested. |
| 19 | CFTIF2 | Channel 2 Transmit/receive FIFO Transmit Interrupt Status Flag<br>0: Transmit/receive FIFO transmit interrupt is not requested.<br>1: Transmit/receive FIFO transmit interrupt is requested. |
| 18 | TQIF2 | Channel 2 Transmit Queue Interrupt Status Flag<br>0: Transmit queue interrupt is not requested.<br>1: Transmit queue interrupt is requested. |

Table 21.112 RSCFDnCFDGTINTSTS0 Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 17 | TAIF2 | Channel 2 Transmit Buffer Abort Interrupt Status Flag<br>0: Transmit buffer abort interrupt is not requested.<br>1: Transmit buffer abort interrupt is requested. |
| 16 | TSIF2 | Channel 2 Transmit Buffer Interrupt Status Flag<br>0: Transmit buffer transmit complete interrupt is not requested.<br>1: Transmit buffer transmit complete interrupt is requested. |
| 15 to 13 | Reserved | When read, the value after reset is returned. |
| 12 | THIF1 | Channel 1 Transmit History Interrupt Status Flag<br>0: Transmit history interrupt is not requested.<br>1: Transmit history interrupt is requested. |
| 11 | CFTIF1 | Channel 1 Transmit/receive FIFO Transmit Interrupt Status Flag<br>0: Transmit/receive FIFO transmit interrupt is not requested.<br>1: Transmit/receive FIFO transmit interrupt is requested. |
| 10 | TQIF1 | Channel 1 Transmit Queue Interrupt Status Flag<br>0: Transmit queue interrupt is not requested.<br>1: Transmit queue interrupt is requested. |
| 9 | TAIF1 | Channel 1 Transmit Buffer Abort Interrupt Status Flag<br>0: Transmit buffer abort interrupt is not requested.<br>1: Transmit buffer abort interrupt is requested. |
| 8 | TSIF1 | Channel 1 Transmit Buffer Interrupt Status Flag<br>0: Transmit buffer transmit complete interrupt is not requested.<br>1: Transmit buffer transmit complete interrupt is requested. |
| 7 to 5 | Reserved | When read, the value after reset is returned. |
| 4 | THIF0 | Channel 0 Transmit History Interrupt Status Flag<br>0: Transmit history interrupt is not requested.<br>1: Transmit history interrupt is requested. |
| 3 | CFTIF0 | Channel 0 Transmit/receive FIFO Transmit Interrupt Status Flag<br>0: Transmit/receive FIFO transmit interrupt is not requested.<br>1: Transmit/receive FIFO transmit interrupt is requested. |
| 2 | TQIF0 | Channel 0 Transmit Queue Interrupt Status Flag<br>0: Transmit queue interrupt is not requested.<br>1: Transmit queue interrupt is requested. |
| 1 | TAIF0 | Channel 0 Transmit Buffer Abort Interrupt Status Flag<br>0: Transmit buffer abort interrupt is not requested.<br>1: Transmit buffer abort interrupt is requested. |
| 0 | TSIF0 | Channel 0 Transmit Buffer Interrupt Status Flag<br>0: Transmit buffer transmit complete interrupt is not requested.<br>1: Transmit buffer transmit complete interrupt is requested. |

### TSIFm Bits

The TSIFm bit is set to 1 when the TMIEp bit in the RSCFDnCFDTMIECy register is set to 1 (transmit buffer interrupt enabled) and the TMTRF[1:0] flags in the RSCFDnCFDTMSTSp register are set to $10_B$ (transmit completed without abort request) or $11_B$ (transmit completed with abort request).

When the TMTRF[1:0] flags are cleared to $00_B$ under the condition that the TSIFm bit can be set to 1, this flag is cleared to 0. In addition, clearing the TMIEp bit to 0 also clears this flag to 0.

### TAIFm Bits

The TAIFm bit is set to 1 when the TAIE bit in the RSCFDnCFDCmCTR register is 1 (transmit abort interrupt enabled) and the TMTRF[1:0] flags in the RSCFDnCFDTMSTSp register are set to $01_B$ (transmit abort completed).

This flag is cleared to 0 when the TMTRF[1:0] flags are cleared to $00_B$ after the transmit abort is completed.

### TQIFm Bits

When the TXQIE bit in the RSCFDnCFDTXQCCm register is set to 1 (transmit queue interrupt enabled) and the TXQIF bit in the RSCFDnCFDTXQSTSm register is set to 1 (transmit queue interrupt request), the TQIFm bit is set to 1.

When the TXQIF bit (transmit queue interrupt request) in the RSCFDnCFDTXQSTSm register is cleared to 0, this bit is cleared to 0. This flag is also cleared to 0 when the TXQIE bit is cleared to 0.

### CFTIFm Bits

When the CFTXIE bit in the RSCFDnCFDCFCCk register is set to 1 (transmit/receive FIFO transmit interrupt enabled) and the CFTXIF bit in the RSCFDnCFDCFSTSk register is set to 1 (transmit/receive FIFO transmit interrupt request), the CFTIFm bit is set to 1.

When the CFTXIF bit is cleared to 0 under the conditions that the CFTIFm bit can be set to 1, this bit is cleared to 0. This flag is also cleared to 0 when the CFTXIE bit is cleared to 0.

### THIFm Bits

When the THLIE bit in the RSCFDnCFDTHLCCm register is set to 1 (transmit history interrupt enabled) and the THLIF bit in the RSCFDnCFDTHLSTSm register is set to 1 (transmit history interrupt request), the THIFm bit is set to 1.

When the THLIF bit in the RSCFDnCFDTHLSTSm register is cleared to 0, this bit is cleared to 0. This flag is also cleared to 0 when the THLIE bit is cleared to 0.

### 21.5.4.7 RSCFDnCFDGTINTSTS1 — Global TX Interrupt Status Register 1

**Access:** RSCFDnCFDGTINTSTS1 register is a read-only register that can be read in 32-bit units
RSCFDnCFDGTINTSTS1L register is a read-only register that can be read in 16-bit units
RSCFDnCFDGTINTSTS1LL, RSCFDnCFDGTINTSTS1LH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDGTINTSTS1: <RSCANn_base> + $0464_H$

RSCFDnCFDGTINTSTS1L: <RSCANn_base> + $0464_H$

RSCFDnCFDGTINTSTS1LL: <RSCANn_base> + $0464_H$,
RSCFDnCFDGTINTSTS1LH: <RSCANn_base> + $0465_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by any reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | THIF5 | CFTIF5 | TQIF5 | TAIF5 | TSIF5 | — | — | — | THIF4 | CFTIF4 | TQIF4 | TAIF4 | TSIF4 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R*1 | R*1 | R*1 | R*1 | R*1 | R | R | R | R*1 | R*1 | R*1 | R*1 | R*1 |

Note 1. This bit is automatically cleared in the global reset or channel reset mode.

**Table 21.113 RSCFDnCFDGTINTSTS1 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 13 | Reserved | When read, the value after reset is returned. |
| 12 | THIF5 | Channel 5 Transmit History Interrupt Status Flag<br>0: Transmit history interrupt is not requested.<br>1: Transmit history interrupt is requested. |
| 11 | CFTIF5 | Channel 5 Transmit/receive FIFO Transmit Interrupt Status Flag<br>0: Transmit/receive FIFO transmit interrupt is not requested.<br>1: Transmit/receive FIFO transmit interrupt is requested. |
| 10 | TQIF5 | Channel 5 Transmit Queue Interrupt Status Flag<br>0: Transmit queue interrupt is not requested.<br>1: Transmit queue interrupt is requested. |
| 9 | TAIF5 | Channel 5 Transmit Buffer Abort Interrupt Status Flag<br>0: Transmit buffer abort interrupt is not requested.<br>1: Transmit buffer abort interrupt is requested. |
| 8 | TSIF5 | Channel 5 Transmit Buffer Interrupt Status Flag<br>0: Transmit buffer transmit complete interrupt is not requested.<br>1: Transmit buffer transmit complete interrupt is requested. |
| 7 to 5 | Reserved | When read, the value after reset is returned. |
| 4 | THIF4 | Channel 4 Transmit History Interrupt Status Flag<br>0: Transmit history interrupt is not requested.<br>1: Transmit history interrupt is requested. |
| 3 | CFTIF4 | Channel 4 Transmit/receive FIFO Transmit Interrupt Status Flag<br>0: Transmit/receive FIFO transmit interrupt is not requested.<br>1: Transmit/receive FIFO transmit interrupt is requested. |
| 2 | TQIF4 | Channel 4 Transmit Queue Interrupt Status Flag<br>0: Transmit queue interrupt is not requested.<br>1: Transmit queue interrupt is requested. |
| 1 | TAIF4 | Channel 4 Transmit Buffer Abort Interrupt Status Flag<br>0: Transmit buffer abort interrupt is not requested.<br>1: Transmit buffer abort interrupt is requested. |

Table 21.113 RSCFDnCFDGTINTSTS1 Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | TSIF4 | Channel 4 Transmit Buffer Interrupt Status Flag<br>0: Transmit buffer transmit complete interrupt is not requested.<br>1: Transmit buffer transmit complete interrupt is requested. |

### TSIFm Bits

The TSIFm bit is set to 1 when the TMIEp bit in the RSCFDnCFDTMIECy register is set to 1 (transmit buffer interrupt enabled) and the TMTRF[1:0] flags in the RSCFDnCFDTMSTSp register are set to $10_B$ (transmit completed without abort request) or $11_B$ (transmit completed with abort request).

When all TMTRF[1:0] flags are cleared to $00_B$ under the condition that the TSIFm bit can be set to 1, this flag is cleared to 0. In addition, clearing the TMIEp bit to 0 also clears this flag to 0.

### TAIFm Bits

The TAIFm bit is set to 1 when the TAIE bit in the RSCFDnCFDCmCTR register is set to 1 (transmit abort interrupt enabled) and the TMTRF[1:0] flags in the RSCFDnCFDTMSTSp register are set to $01_B$ (transmit abort completed).

This flag is cleared to 0 when all TMTRF[1:0] flags are cleared to $00_B$ after the transmit abort is completed.

### TQIFm Bits

When the TXQIE bit in the RSCFDnCFDTXQCCm register is set to 1 (transmit queue interrupt enabled) and the TXQIF bit in the RSCFDnCFDTXQSTSm register is set to 1 (transmit queue interrupt request), the TQIFm bit is set to 1.

When the TXQIF bit (transmit queue interrupt request) in the RSCFDnCFDTXQSTSm register is cleared to 0, this bit is cleared to 0. Clearing the TXQIE bit to 0 also clears this flag to 0.

### CFTIFm Bits

When the CFTXIE bit in the RSCFDnCFDCFCCk register is set to 1 (transmit/receive FIFO transmit interrupt enabled) and the CFTXIF bit in the RSCFDnCFDCFSTSk register is set to 1 (transmit/receive FIFO transmit interrupt request), the CFTIFm bit is set to 1.

When the CFTXIF bit is cleared to 0 under the conditions that the CFTIFm bit can be set to 1, this bit is cleared to 0. This flag is also cleared to 0 when the CFTXIE bit is cleared to 0.

### THIFm Bits

When the THLIE bit in the RSCFDnCFDTHLCCm register is set to 1 (transmit history interrupt enabled) and the THLIF bit in the RSCFDnCFDTHLSTSm register is set to 1 (transmit history interrupt request), the THIFm bit is set to 1.

When the THLIF bit in the RSCFDnCFDTHLSTSm register is cleared to 0, this bit is cleared to 0.

This flag is also cleared to 0 when the THLIE bit is cleared to 0.

### 21.5.4.8 RSCFDnCFDGFDCFG — Global FD Configuration Register

**Access:** RSCFDnCFDGFDCFG register can be read or written in 32-bit unit.
RSCFDnCFDGFDCFGL register can be read or written in 16-bit unit.
RSCFDnCFDGFDCFGLL, RSCFDnCFDGFDCFGLH registers can be read or written in 8-bit unit.

**Address:** RSCFDnCFDGFDCFG: <RSCANn_base> + $0474_H$

RSCFDnCFDGFDCFGL: <RSCANn_base> + $0474_H$

RSCFDnCFDGFDCFGLL: <RSCANn_base> + $0474_H$,
RSCFDnCFDGFDCFGLH: <RSCANn_base> + $0475_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TSCCFG[1:0] | | — | — | — | — | — | — | — | PRED |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R | R/W |

**Table 21.114 RSCFDnCFDGFDCFG Register Content**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 10 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 9, 8 | TSCCFG[1:0] | Timestamp Capture Setting<br>B9 b8<br>0 0: Captured at a sample point in the SOF bit.<br>0 1: Captured when a valid frame has been transmitted/received.<br>1 0: Captured at a sample point of the res bit.*1<br>1 1: Setting prohibited |
| 7 to 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | PRED | Protocol Exception Event Detection Disable<br>0: Protocol exception event detection enabled<br>1: Protocol exception event detection disabled |

Note 1. When a Classical CAN frame is transmitted/received, a timestamp value will be captured at the sample point in the SOF bit.

#### TSCCFG[1:0] Bits

Select a point where a timestamp value is captured. Modify this bit only in global reset mode.

#### PRED Bit

Setting this bit to 1 disables the protocol exception event detection. When a protocol exception event is detected while this bit is set to 1, the event is regarded as a form error and an error frame will be output. Modify this bit only in global reset mode.

## 21.5.5 Details of Receive Rule-related Registers

### 21.5.5.1 RSCFDnCFDGAFLECTR — Receive Rule Entry Control Register

**Access:** RSCFDnCFDGAFLECTR register can be read or written in 32-bit units
RSCFDnCFDGAFLECTRL register can be read or written in 16-bit units
RSCFDnCFDGAFLECTRLL, RSCFDnCFDGAFLECTRLH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDGAFLECTR: <RSCANn_base> + $0098_H$

RSCFDnCFDGAFLECTRL: <RSCANn_base> + $0098_H$

RSCFDnCFDGAFLECTRLL: <RSCANn_base> + $0098_H$,
RSCFDnCFDGAFLECTRLH: <RSCANn_base> + $0099_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | AFLDAE | — | — | — | AFLPN[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 21.115 RSCFDnCFDGAFLECTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 8 | AFLDAE | Receive Rule Table Write Enable<br>0: Receive rule table write is disabled.<br>1: Receive rule table write is enabled. |
| 7 to 5 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 4 to 0 | AFLPN[4:0] | Receive Rule Table Page Number Configuration<br>A page number can be selected from the range of page 0 ($00000_B$) to page 23 ($10111_B$). |

#### AFLDAE Bit

Setting this bit to 0 disables the write to the receive rule table. After writes to the receive rule table are completed, set this bit to 0 to disable the write to the table. The receive rule table can be read regardless of the value of this bit.

Set the AFLDAE bit to 1 only in global reset mode.

#### AFLPN[4:0] Bits

These bits are used to set the page number of the receive rule table. Sixteen receive rules can be set per page.

Set these bits to a value within the range of $00000_B$ to $10111_B$.

### 21.5.5.2 RSCFDnCFDGAFLCFG0 — Receive Rule Configuration Register 0

**Access:** RSCFDnCFDGAFLCFG0 register can be read or written in 32-bit units
RSCFDnCFDGAFLCFG0L, RSCFDnCFDGAFLCFG0H registers can be read or written in 16-bit units
RSCFDnCFDGAFLCFG0LL, RSCFDnCFDGAFLCFG0LH, RSCFDnCFDGAFLCFG0HL,
RSCFDnCFDGAFLCFG0HH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDGAFLCFG0: <RSCANn_base> + $009C_H$

RSCFDnCFDGAFLCFG0L: <RSCANn_base> + $009C_H$,
RSCFDnCFDGAFLCFG0H: <RSCANn_base> + $009E_H$

RSCFDnCFDGAFLCFG0LL: <RSCANn_base> + $009C_H$,
RSCFDnCFDGAFLCFG0LH: <RSCANn_base> + $009D_H$,
RSCFDnCFDGAFLCFG0HL: <RSCANn_base> + $009E_H$,
RSCFDnCFDGAFLCFG0HH: <RSCANn_base> + $009F_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RNC0[7:0] | | | | | | | | RNC1[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RNC2[7:0] | | | | | | | | RNC3[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.116 RSCFDnCFDGAFLCFG0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RNC0[7:0] | Number of Rules for Channel 0<br>Set the number of receive rules exclusively used for channel 0. |
| 23 to 16 | RNC1[7:0] | Number of Rules for Channel 1<br>Set the number of receive rules exclusively used for channel 1. |
| 15 to 8 | RNC2[7:0] | Number of Rules for Channel 2<br>Set the number of receive rules exclusively used for channel 2. |
| 7 to 0 | RNC3[7:0] | Number of Rules for Channel 3<br>Set the number of receive rules exclusively used for channel 3. |

Modify the RSCFDnCFDGAFLCFG0 register only in global reset mode.

Up to 64 × (number of channels) rules can be registered in the receive rule table as the entire unit. The number of receive rules per channel should meet the following conditions.

- The maximum number of rules per channel is 128.
- The total of the number of rules allocated to each channel is not larger than the number of rules that can be registered in the entire unit.

#### RNC0[7:0] Bits

These bits are used to set the number of rules to be registered in the channel 0 receive rule table.

Set these bits to a value within the range of $00_H$ to $80_H$.

#### RNC1[7:0] Bits

These bits are used to set the number of rules to be registered in the channel 1 receive rule table.

Set these bits to a value within the range of $00_H$ to $80_H$.

**RNC2[7:0] Bits**

These bits are used to set the number of rules to be registered in the channel 2 receive rule table.

Set these bits to a value within the range of $00_H$ to $80_H$.

**RNC3[7:0] Bits**

These bits are used to set the number of rules to be registered in the channel 3 receive rule table.

Set these bits to a value within the range of $00_H$ to $80_H$.

### 21.5.5.3 RSCFDnCFDGAFLCFG1 — Receive Rule Configuration Register 1

**Access:** RSCFDnCFDGAFLCFG1 register can be read or written in 32-bit units
RSCFDnCFDGAFLCFG1H register can be read or written in 16-bit units
RSCFDnCFDGAFLCFG1HL, RSCFDnCFDGAFLCFG1HH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDGAFLCFG1: <RSCANn_base> + $00A0_H$

RSCFDnCFDGAFLCFG1H: <RSCANn_base> + $00A2_H$

RSCFDnCFDGAFLCFG1HL: <RSCANn_base> + $00A2_H$,
RSCFDnCFDGAFLCFG1HH: <RSCANn_base> + $00A3_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RNC4[7:0] | | | | | | | | RNC5[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.117 RSCFDnCFDGAFLCFG1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RNC4[7:0] | Number of Rules for Channel 4<br>Set the number of receive rules exclusively used for channel 4. |
| 23 to 16 | RNC5[7:0] | Number of Rules for Channel 5<br>Set the number of receive rules exclusively used for channel 5. |
| 15 to 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Modify the RSCFDnCFDGAFLCFG1 register only in global reset mode.

Up to 64 × (number of channels) rules can be registered in the receive rule table as the entire unit. The number of receive rules per channel should meet the following conditions.

- The maximum number of rules per channel is 128.
- The total of the number of rules allocated to each channel is not larger than the number of rules that can be registered in the entire unit.

#### RNC4[7:0] Bits

These bits are used to set the number of rules to be registered in the channel 4 receive rule table.

Set these bits to a value within the range of $00_H$ to $80_H$.

#### RNC5[7:0] Bits

These bits are used to set the number of rules to be registered in the channel 5 receive rule table.

Set these bits to a value within the range of $00_H$ to $80_H$.

### 21.5.5.4 RSCFDnCFDGAFLIDj — Receive Rule ID Register (j = 0 to 15)

**Access:** RSCFDnCFDGAFLIDj register can be read or written in 32-bit units
RSCFDnCFDGAFLIDjL, RSCFDnCFDGAFLIDjH registers can be read or written in 16-bit units
RSCFDnCFDGAFLIDjLL, RSCFDnCFDGAFLIDjLH, RSCFDnCFDGAFLIDjHL, RSCFDnCFDGAFLIDjHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDGAFLIDj: <RSCANn_base> + $1000_H$ + ($10_H$ × j)

RSCFDnCFDGAFLIDjL: <RSCANn_base> + $1000_H$ + ($10_H$ × j),
RSCFDnCFDGAFLIDjH: <RSCANn_base> + $1002_H$ + ($10_H$ × j)

RSCFDnCFDGAFLIDjLL: <RSCANn_base> + $1000_H$ + ($10_H$ × j),
RSCFDnCFDGAFLIDjLH: <RSCANn_base> + $1001_H$ + ($10_H$ × j),
RSCFDnCFDGAFLIDjHL: <RSCANn_base> + $1002_H$ + ($10_H$ × j),
RSCFDnCFDGAFLIDjHH: <RSCANn_base> + $1003_H$ + ($10_H$ × j)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLIDE | GAFLRTR | GAFLLB | GAFLID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.118 RSCFDnCFDGAFLIDj Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | GAFLIDE | IDE Select<br>0: Standard ID<br>1: Extended ID |
| 30 | GAFLRTR | RTR Select<br>0: Data frame<br>1: Remote frame |
| 29 | GAFLLB | Receive Rule Target Message Select<br>0: When a message transmitted from another CAN node is received<br>1: When the own transmitted message is received |
| 28 to 0 | GAFLID[28:0] | ID<br>Set the ID of the receive rule.<br>For the standard ID, set the ID in bits b10 to b0 and set bits b28 to b11 to 0. |

Modify the RSCFDnCFDGAFLIDj register when the AFLDAE bit in the RSCFDnCFDGAFLECTR register is set to 1 (receive rule table write is enabled) in global reset mode.

#### GAFLIDE Bit

This bit is used to select the ID format (standard ID or extended ID) of the receive rule. This bit is compared with the IDE bit in the received message during the acceptance filter processing.

#### GAFLRTR Bit

This bit is used to select the frame format (data frame or remote frame) of the receive rule. This bit is compared with the RTR bit in the received message during the acceptance filter processing.

#### GAFLLB Bit

When this bit is set to 0, data processing using the receive rule is performed when receiving messages transmitted from another CAN node.

When this bit is set to 1 when the mirror function is used, data processing using the receive rule is performed when the CAN node is receiving its own transmitted messages.

#### GAFLID[28:0] Bits

These bits are used to set the ID field of the receive rule. The ID value set by these bits is compared with the ID of the received message during the acceptance filter processing.

### 21.5.5.5 RSCFDnCFDGAFLMj — Receive Rule Mask Register (j = 0 to 15)

**Access:** RSCFDnCFDGAFLMj register can be read or written in 32-bit units
RSCFDnCFDGAFLMjL, RSCFDnCFDGAFLMjH registers can be read or written in 16-bit units
RSCFDnCFDGAFLMjLL, RSCFDnCFDGAFLMjLH, RSCFDnCFDGAFLMjHL, RSCFDnCFDGAFLMjHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDGAFLMj: <RSCANn_base> + $1004_H$ + ($10_H$ × j)

RSCFDnCFDGAFLMjL: <RSCANn_base> + $1004_H$ + ($10_H$ × j),
RSCFDnCFDGAFLMjH: <RSCANn_base> + $1006_H$ + ($10_H$ × j)

RSCFDnCFDGAFLMjLL: <RSCANn_base> + $1004_H$ + ($10_H$ × j),
RSCFDnCFDGAFLMjLH: <RSCANn_base> + $1005_H$ + ($10_H$ × j),
RSCFDnCFDGAFLMjHL: <RSCANn_base> + $1006_H$ + ($10_H$ × j),
RSCFDnCFDGAFLMjHH: <RSCANn_base> + $1007_H$ + ($10_H$ × j)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLIDEM | GAFLRTRM | — | GAFLIDM[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLIDM[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.119 RSCFDnCFDGAFLMj Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | GAFLIDEM | IDE Mask<br>0: The IDE bit is not compared.<br>1: The IDE bit is compared. |
| 30 | GAFLRTRM | RTR Mask<br>0: The RTR bit is not compared.<br>1: The RTR bit is compared |
| 29 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 28 to 0 | GAFLIDM[28:0] | ID Mask<br>0: The corresponding ID bit is not compared.<br>1: The corresponding ID bit is compared. |

Modify the RSCFDnCFDGAFLMj register when the AFLDAE bit in the RSCFDnCFDGAFLECTR register is set to 1 (receive rule table write is enabled) in global reset mode.

#### GAFLIDEM Bit

When this bit is set to 1, filter processing is performed only for messages of the ID format specified by the GAFLIDE bit in the RSCFDnCFDGAFLIDj register.

When this bit is cleared to 0, the IDs of all the receive messages and the specified IDs are regarded as matched. To set the GAFLIDEM bit to 0, set the GAFLIDM[28:0] bits to all 0 at the same time.

#### GAFLRTRM Bit

This bit is used to mask the RTR bit of the receive rule.

#### GAFLIDM[28:0] Bits

These bits are used to mask the corresponding ID bit of the receive rule.

### 21.5.5.6 RSCFDnCFDGAFLP0_j — Receive Rule Pointer 0 Register (j = 0 to 15)

**Access:** RSCFDnCFDGAFLP0_j register can be read or written in 32-bit units
RSCFDnCFDGAFLP0_jL, RSCFDnCFDGAFLP0_jH registers can be read or written in 16-bit units
RSCFDnCFDGAFLP0_jLH, RSCFDnCFDGAFLP0_jHL, RSCFDnCFDGAFLP0_jHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDGAFLP0_j: <RSCANn_base> + $1008_H$ + ($10_H$ × j)

RSCFDnCFDGAFLP0_jL: <RSCANn_base> + $1008_H$ + ($10_H$ × j),
RSCFDnCFDGAFLP0_jH: <RSCANn_base> + $100A_H$ + ($10_H$ × j)

RSCFDnCFDGAFLP0_jLH: <RSCANn_base> + $1009_H$ + ($10_H$ × j),
RSCFDnCFDGAFLP0_jHL: <RSCANn_base> + $100A_H$ + ($10_H$ × j),
RSCFDnCFDGAFLP0_jHH: <RSCANn_base> + $100B_H$ + ($10_H$ × j)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLDLC[3:0] | | | | GAFLPTR[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLRMV | GAFLRMDP[6:0] | | | | | | | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

**Table 21.120 RSCFDnCFDGAFLP0_j Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | GAFLDLC[3:0] | Receive Rule DLC |

| b31 | b30 | b29 | b28 | Classical CAN Frame | CAN FD Frame |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 data bytes | |
| 0 | 0 | 0 | 1 | 1 data byte | |
| 0 | 0 | 1 | 0 | 2 data bytes | |
| 0 | 0 | 1 | 1 | 3 data bytes | |
| 0 | 1 | 0 | 0 | 4 data bytes | |
| 0 | 1 | 0 | 1 | 5 data bytes | |
| 0 | 1 | 1 | 0 | 6 data bytes | |
| 0 | 1 | 1 | 1 | 7 data bytes | |
| 1 | 0 | 0 | 0 | 8 data bytes | |
| 1 | 0 | 0 | 1 | 8 data bytes | 12 data bytes |
| 1 | 0 | 1 | 0 | | 16 data bytes |
| 1 | 0 | 1 | 1 | | 20 data bytes |
| 1 | 1 | 0 | 0 | | 24 data bytes |
| 1 | 1 | 0 | 1 | | 32 data bytes |
| 1 | 1 | 1 | 0 | | 48 data bytes |
| 1 | 1 | 1 | 1 | | 64 data bytes |

| | | |
|---|---|---|
| 27 to 16 | GAFLPTR[11:0] | Receive Rule Label<br>Set the 12-bit label information. |
| 15 | GAFLRMV | Receive Buffer Enable<br>0: No receive buffer is used.<br>1: A receive buffer is used. |
| 14 to 8 | GAFLRMDP[6:0] | Receive Buffer Number Select<br>Set the receive buffer number to store receive messages. |

Table 21.120 RSCFDnCFDGAFLP0_j Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Modify the RSCFDnCFDGAFLP0_j register when the AFLDAE bit in the RSCFDnCFDGAFLECTR register is set to 1 (receive rule table write is enabled) in global reset mode.

### GAFLDLC[3:0] Bits

These bits are used to set the minimum data length necessary for receiving messages. If the data length of a message that is being filtered is equal to or larger than the value set by the GAFLDLC[3:0] bits, the message passes the DLC check. Setting these bits to $0000_B$ disables the DLC check function allowing messages with any data length to pass the DLC check.

### GAFLPTR[11:0] Bits

These bits are used to set a 12-bit label to be attached to messages that have passed through the filter. A label is attached when a message is stored in the receive buffer or the FIFO buffer.

### GAFLRMV Bit

When this bit is set to 1, receive messages that have passed through the filter are stored in the receive buffer selected by the GAFLRMDP[6:0] bits.

### GAFLRMDP[6:0] Bits

These bits are used to select the number of the receive buffer that stores receive messages that have passed through the filter when the GAFLRMV bit is set to 1. Set these bits to a value smaller than the value set by the NRXMB[7:0] bits in the RSCFDnCFDRMNB register.

### 21.5.5.7 RSCFDnCFDGAFLP1_j — Receive Rule Pointer 1 Register (j = 0 to 15)

**Access:** RSCFDnCFDGAFLP1_j register can be read or written in 32-bit units
RSCFDnCFDGAFLP1_jL, RSCFDnCFDGAFLP1_jH registers can be read or written in 16-bit units
RSCFDnCFDGAFLP1_jLL, RSCFDnCFDGAFLP1_jLH, RSCFDnCFDGAFLP1_jHL,
RSCFDnCFDGAFLP1_jHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDGAFLP1_j: <RSCANn_base> + $100C_H$ + ($10_H$ × j)

RSCFDnCFDGAFLP1_jL: <RSCANn_base> +$100C_H$ + ($10_H$ × j),
RSCFDnCFDGAFLP1_jH: <RSCANn_base> + $100E_H$ + ($10_H$ × j)

RSCFDnCFDGAFLP1_jLL: <RSCANn_base> + $100C_H$ + ($10_H$ × j),
RSCFDnCFDGAFLP1_jLH: <RSCANn_base> + $100D_H$ + ($10_H$ × j),
RSCFDnCFDGAFLP1_jHL: <RSCANn_base> + $100E_H$ + ($10_H$ × j),
RSCFDnCFDGAFLP1_jHH: <RSCANn_base> + $100F_H$ + ($10_H$ × j)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | GAFLFDP[25:16] | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLFDP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.121 RSCFDnCFDGAFLP1_j Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 25 to 8 | GAFLFDP[25:8] | Transmit/Receive FIFO Buffer k Select<br>(Bit position –8 = target transmit/receive FIFO buffer number k)<br>0: Transmit/receive FIFO buffer is not selected.<br>1: Transmit/receive FIFO buffer is selected. |
| 7 to 0 | GAFLFDP[7:0] | Receive FIFO Buffer x Select<br>(Bit position = target receive FIFO buffer number x)<br>0: Receive FIFO buffer is not selected.<br>1: Receive FIFO buffer is selected. |

Modify the RSCFDnCFDGAFLP1_j register when the AFLDAE bit in the RSCFDnCFDGAFLECTR register is set to 1 (receive rule table write is enabled) in global reset mode.

#### GAFLFDP[25:0] Bits

These bits are used to specify FIFO buffers that store receive messages that have passed through the filter. Up to eight FIFO buffers can be selected. However, when the GAFLRMV bit in the RSCFDnCFDGAFLP0_j register is set to 1 (a message is stored in the receive buffer), up to seven FIFO buffers can be selected. Only receive FIFO buffers and the transmit/receive FIFO buffer for which the CFM[1:0] bits in the RSCFDnCFDCFCCk register are set to $00_B$ (receive mode) or $10_B$ (gateway mode) can be selected.

## 21.5.6 Details of Receive Buffer-related Registers

### 21.5.6.1 RSCFDnCFDRMNB — Receive Buffer Number Register

**Access:** RSCFDnCFDRMNB register can be read or written in 32-bit units
RSCFDnCFDRMNBL register can be read or written in 16-bit units
RSCFDnCFDRMNBLL, RSCFDnCFDRMNBLH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDRMNB: <RSCANn_base> + $00A4_H$

RSCFDnCFDRMNBL: <RSCANn_base> + $00A4_H$

RSCFDnCFDRMNBLL: <RSCANn_base> + $00A4_H$,
RSCFDnCFDRMNBLH: <RSCANn_base> + $00A5_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | RMPLS[1:0] | | NRXMB[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.122 RSCFDnCFDRMNB Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 10 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 9, 8 | RMPLS[1:0] | Receive Buffer Payload Storage Size Select<br>b9 b8<br>0 0: 8 bytes<br>0 1: 12 bytes<br>1 0: 16 bytes<br>1 1: 20 bytes |
| 7 to 0 | NRXMB[7:0] | Receive Buffer Number Configuration<br>Set the number of receive buffers.<br>Set a value of 0 to 96. |

Modify the RSCFDnCFDRMNB register only in global reset mode.

#### RMPLS[1:0] Bits

These bits are used to select the maximum payload size that can be stored in the receive buffer.

#### NRXMB[7:0] Bits

These bits are used to set the total number of receive buffers of the RS-CANFD module. The maximum value is 16 × (number of channels).

Setting all of these bits to 0 makes receive buffers unavailable.

### 21.5.6.2 RSCFDnCFDRMNDy — Receive Buffer New Data Register (y = 0 to 2)

**Access:** RSCFDnCFDRMNDy register can be read or written in 32-bit units
RSCFDnCFDRMNDyL, RSCFDnCFDRMNDyH registers can be read or written in 16-bit units
RSCFDnCFDRMNDyLL, RSCFDnCFDRMNDyLH, RSCFDnCFDRMNDyHL, RSCFDnCFDRMNDyHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDRMNDy: <RSCANn_base> + $00A8_H$ + ($04_H$ × y)

RSCFDnCFDRMNDyL: <RSCANn_base> + $00A8_H$ + ($04_H$ × y),
RSCFDnCFDRMNDyH: <RSCANn_base> + $00AA_H$ + ($04_H$ × y)

RSCFDnCFDRMNDyLL: <RSCANn_base> + $00A8_H$ + ($04_H$ × y),
RSCFDnCFDRMNDyLH: <RSCANn_base> + $00A9_H$ + ($04_H$ × y),
RSCFDnCFDRMNDyHL: <RSCANn_base> + $00AA_H$ + ($04_H$ × y),
RSCFDnCFDRMNDyHH: <RSCANn_base> + $00AB_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMNSq (q = y × 32 + 31 to y × 32 + 16 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMNSq (q = y × 32 + 15 to y × 32 + 0 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.123 RSCFDnCFDRMNDy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | RMNSq | Receive Buffer Receive Complete Flag q (q = y × 32 + 31 to y × 32 + 16)<br>0: There is no new message in receive buffer q.<br>1: There is a new message in receive buffer q. |
| 15 to 0 | RMNSq | Receive Buffer Receive Complete Flag q (q = y × 32 + 15 to y × 32 + 0)<br>0: There is no new message in receive buffer q.<br>1: There is a new message in receive buffer q. |

Write 0 to the RSCFDnCFDRMNDy register in global operating mode or global test mode.

#### RMNSq Flags (q = 0 to 95)

Each RMNSq flag is set to 1 when the processing for storing a message in the corresponding receive buffer starts.

To clear a flag to 0, the program must write 0 to the flag. Use a store instruction to write “0” to the flag and “1” to other flags. These bits cannot be set to 0 while a message is being stored. The message storing time depends on the storage payload size of the receive buffer. When the RMPLS[1:0] value in the RSCFDnCFDRMNB register is $00_B$ (8 bytes), the message storing time is 12 pclk clock cycles. When the RMPLS[1:0] value is $11_B$ (20 bytes), the message storing time is 18 pclk clock cycles. (2 pclk clock cycles per 4 bytes of storage payload size).

These flags are cleared to 0 in global reset mode.

### 21.5.6.3 RSCFDnCFDRMIDq — Receive Buffer ID Register (q = 0 to 95)

**Access:** RSCFDnCFDRMIDq register is a read-only register that can be read in 32-bit units
RSCFDnCFDRMIDqL, RSCFDnCFDRMIDqH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDRMIDqLL, RSCFDnCFDRMIDqLH, RSCFDnCFDRMIDqHL, RSCFDnCFDRMIDqHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDRMIDq: <RSCANn_base> + $2000_H$ + ($20_H$ × q)

RSCFDnCFDRMIDqL: <RSCANn_base> + $2000_H$ + ($20_H$ × q),
RSCFDnCFDRMIDqH: <RSCANn_base> + $2002_H$ + ($20_H$ × q)

RSCFDnCFDRMIDqLL: <RSCANn_base> + $2000_H$ + ($20_H$ × q),
RSCFDnCFDRMIDqLH: <RSCANn_base> + $2001_H$ + ($20_H$ × q),
RSCFDnCFDRMIDqHL: <RSCANn_base> + $2002_H$ + ($20_H$ × q),
RSCFDnCFDRMIDqHH: <RSCANn_base> + $2003_H$ + ($20_H$ × q)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMIDE | RMRTR | — | RMID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.124 RSCFDnCFDRMIDq Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | RMIDE | Receive Buffer IDE<br>0: Standard ID<br>1: Extended ID |
| 30 | RMRTR | Receive Buffer RTR/RRS<br>• When the received message is a classical CAN frame<br>0: Data frame<br>1: Remote frame<br>• When the received message is a CAN FD frame<br>The RRS bit value of the received message can be read. |
| 29 | Reserved | When read, the value after reset is returned. |
| 28 to 0 | RMID[28:0] | Receive Buffer ID Data<br>These bits contain the standard ID or extended ID of the received message.<br>Read bits b10 to b0 for standard ID. Bits b28 to b11 are read as 0. |

#### RMIDE Bit

This bit indicates the ID format (standard ID or extended ID) of the message stored in the receive buffer.

#### RMRTR Bit

When the received message is a classical CAN frame, this bit indicates the frame format (data frame or remote frame) of the message stored in the receive buffer. When the received message is a CAN FD frame, this bit indicates the RRS bit value in the message.

#### RMID[28:0] Bits

These bits contain the ID of the message stored in the receive buffer.

### 21.5.6.4 RSCFDnCFDRMPTRq — Receive Buffer Pointer Register (q = 0 to 95)

**Access:** RSCFDnCFDRMPTRq register is a read-only register that can be read in 32-bit units
RSCFDnCFDRMPTRqL, RSCFDnCFDRMPTRqH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDRMPTRqLL, RSCFDnCFDRMPTRqLH, RSCFDnCFDRMPTRqHL, RSCFDnCFDRMPTRqHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDRMPTRq: <RSCANn_base> + $2004_H$ + ($20_H$ × q)

RSCFDnCFDRMPTRqL: <RSCANn_base> + $2004_H$ + ($20_H$ × q),
RSCFDnCFDRMPTRqH: <RSCANn_base> + $2006_H$ + ($20_H$ × q)

RSCFDnCFDRMPTRqLL: <RSCANn_base> + $2004_H$ + ($20_H$ × q),
RSCFDnCFDRMPTRqLH: <RSCANn_base> + $2005_H$ + ($20_H$ × q),
RSCFDnCFDRMPTRqHL: <RSCANn_base> + $2006_H$ + ($20_H$ × q),
RSCFDnCFDRMPTRqHH: <RSCANn_base> + $2007_H$ + ($20_H$ × q)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMDLC[3:0] | | | | RMPTR[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMTS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.125 RSCFDnCFDRMPTRq Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | RMDLC[3:0] | Receive Buffer DLC Data |
| 27 to 16 | RMPTR[11:0] | Receive Buffer Label Data<br>Label information of the received message. |
| 15 to 0 | RMTS[15:0] | Receive Buffer Timestamp Data<br>Timestamp value of the received message. |

| b31 | b30 | b29 | b28 | Classical CAN Frame | CAN FD Frame |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 data bytes | |
| 0 | 0 | 0 | 1 | 1 data byte | |
| 0 | 0 | 1 | 0 | 2 data bytes | |
| 0 | 0 | 1 | 1 | 3 data bytes | |
| 0 | 1 | 0 | 0 | 4 data bytes | |
| 0 | 1 | 0 | 1 | 5 data bytes | |
| 0 | 1 | 1 | 0 | 6 data bytes | |
| 0 | 1 | 1 | 1 | 7 data bytes | |
| 1 | 0 | 0 | 0 | 8 data bytes | |
| 1 | 0 | 0 | 1 | 8 data bytes | 12 data bytes |
| 1 | 0 | 1 | 0 | | 16 data bytes |
| 1 | 0 | 1 | 1 | | 20 data bytes |
| 1 | 1 | 0 | 0 | | 24 data bytes |
| 1 | 1 | 0 | 1 | | 32 data bytes |
| 1 | 1 | 1 | 0 | | 48 data bytes |
| 1 | 1 | 1 | 1 | | 64 data bytes |

#### RMDLC[3:0] Bits

These bits indicate the data length of the message stored in the receive buffer. The number of bytes of the payload to be stored in the receive buffer is determined by the RMPLS[1:0] bits in the RSCFDnCFDRMNB register.

#### RMPTR[11:0] Bits

These bits indicate the label information of the message stored in the receive buffer.

#### RMTS[15:0] Bits

These bits indicate the timestamp value of the message stored in the receive buffer.

### 21.5.6.5 RSCFDnCFDRMFDSTSq — Receive Buffer CAN FD Status Register (q = 0 to 95)

**Access:** RSCFDnCFDRMFDSTSq register is a read-only register that can be read in 32-bit units
RSCFDnCFDRMFDSTSqL register is a read-only register that can be read in 16-bit units
RSCFDnCFDRMFDSTSqLL register is a read-only register that can be read in 8-bit units

**Address:** RSCFDnCFDRMFDSTSq: <RSCANn_base> + $2008_H$ + ($20_H$ × q)

RSCFDnCFDRMFDSTSqL: <RSCANn_base> + $2008_H$ + ($20_H$ × q)

RSCFDnCFDRMFDSTSqLL: <RSCANn_base> + $2008_H$ + ($20_H$ × q)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | RMFDF | RMBRS | RMESI |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.126 RSCFDnCFDRMFDSTSq Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 | RMFDF | FDF<br>0: Classical CAN frame<br>1: CAN FD frame |
| 1 | RMBRS | BRS<br>0: The bit rate in the data area does not change.<br>1: The bit rate in the data area changes. |
| 0 | RMESI | ESI<br>0: Error active node<br>1: Error passive node |

#### RMFDF Bit

This bit indicates the FD format (classical CAN frame or CAN FD frame) of the message stored in the receive buffer.

#### RMBRS Bit

When the RMFDF bit is 1, this bit indicates the BRS bit value of the message stored in the receive buffer. When the RMFDF bit is 0, this bit is always read as 0.

#### RMESI Bit

When the RMFDF bit is 1, this bit indicates the ESI bit value of the message stored in the receive buffer. When the RMFDF bit is 0, this bit is always read as 0.

### 21.5.6.6 RSCFDnCFDRMDFb_q — Receive Buffer Data Field b Register (b = 0 to 4, q = 0 to 95)

**Access:** RSCFDnCFDRMDFb_q register is a read-only register that can be read in 32-bit units
RSCFDnCFDRMDFb_qL, RSCFDnCFDRMDFb_qH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDRMDFb_qLL, RSCFDnCFDRMDFb_qLH, RSCFDnCFDRMDFb_qHL, RSCFDnCFDRMDFb_qHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDRMDFb_q: <RSCANn_base> + $200C_H$ + ($04_H$ × b) + ($20_H$ × q)

RSCFDnCFDRMDFb_qL: <RSCANn_base> + $200C_H$ + ($04_H$ × b) + ($20_H$ × q),
RSCFDnCFDRMDFb_qH: <RSCANn_base> + $200E_H$ + ($04_H$ × b) + ($20_H$ × q)

RSCFDnCFDRMDFb_qLL: <RSCANn_base> + $200C_H$ + ($04_H$ × b) + ($20_H$ × q),
RSCFDnCFDRMDFb_qLH: <RSCANn_base> + $200D_H$ + ($04_H$ × b) + ($20_H$ × q),
RSCFDnCFDRMDFb_qHL: <RSCANn_base> + $200E_H$ + ($04_H$ × b) + ($20_H$ × q),
RSCFDnCFDRMDFb_qHH: <RSCANn_base> + $200F_H$ + ($04_H$ × b) + ($20_H$ × q)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMDB4 × b + 3 [7:0] | | | | | | | | RMDB4 × b + 2 [7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMDB4 × b + 1 [7:0] | | | | | | | | RMDB4 × b + 0 [7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.127 RSCFDnCFDRMDFb_q Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RMDB4 × b + 3 [7:0] | Receive Buffer Data Byte 4 × b + 3<br>Receive Buffer Data Byte 4 × b + 2<br>Receive Buffer Data Byte 4 × b + 1<br>Receive Buffer Data Byte 4 × b + 0 |
| 23 to 16 | RMDB4 × b + 2 [7:0] | |
| 15 to 8 | RMDB4 × b + 1 [7:0] | Data for a message stored in the receive buffer can be read. |
| 7 to 0 | RMDB4 × b + 0 [7:0] | |

When the RMDLC[3:0] value in the RSCFDnCFDRMPTRq register is smaller than the payload storage size of the receive buffer, data bytes for which no data is set are read as $00_H$.

Specify the payload storage size of the receive buffer by the RMPLS[1:0] bits in the RSCFDnCFDRMNB register. Do not read the RSCFDnCFDRMDFb_q register corresponding to an area larger than the specified size.

## 21.5.7 Details of Receive FIFO Buffer-related Registers

### 21.5.7.1 RSCFDnCFDRFCCx — Receive FIFO Buffer Configuration and Control Register (x = 0 to 7)

**Access:** RSCFDnCFDRFCCx register can be read or written in 32-bit units
RSCFDnCFDRFCCxL register can be read or written in 16-bit units
RSCFDnCFDRFCCxLL, RSCFDnCFDRFCCxLH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDRFCCx: <RSCANn_base> + $00B8_H$ + ($04_H$ × x)

RSCFDnCFDRFCCxL: <RSCANn_base> + $00B8_H$ + ($04_H$ × x)

RSCFDnCFDRFCCxLL: <RSCANn_base> + $00B8_H$ + ($04_H$ × x),
RSCFDnCFDRFCCxLH: <RSCANn_base> + $00B9_H$ + ($04_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFIGCV[2:0] | | | RFIM | — | RFDC[2:0] | | | — | RFPLS[2:0] | | | — | — | RFIE | RFE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R | R/W | R/W |

**Table 21.128 RSCFDnCFDRFCCx Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 15 to 13 | RFIGCV[2:0] | Receive FIFO Interrupt Request Timing Select<br>b15 b14 b13<br>0 0 0: When FIFO is 1/8 full.<br>0 0 1: When FIFO is 2/8 full.<br>0 1 0: When FIFO is 3/8 full.<br>0 1 1: When FIFO is 4/8 full.<br>1 0 0: When FIFO is 5/8 full.<br>1 0 1: When FIFO is 6/8 full.<br>1 1 0: When FIFO is 7/8 full.<br>1 1 1: When FIFO is full. |
| 12 | RFIM | Receive FIFO Interrupt Source Select<br>0: An interrupt occurs when the condition set by the RFIGCV[2:0] bits is met.<br>1: An interrupt occurs each time a message has been received. |
| 11 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 10 to 8 | RFDC[2:0] | Receive FIFO Buffer Depth Configuration<br>b10 b9 b8<br>0 0 0: 0 messages<br>0 0 1: 4 messages<br>0 1 0: 8 messages<br>0 1 1: 16 messages<br>1 0 0: 32 messages<br>1 0 1: 48 messages<br>1 1 0: 64 messages<br>1 1 1: 128 messages |
| 7 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Table 21.128 RSCFDnCFDRFCCx Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 6 to 4 | RFPLS[2:0] | Receive FIFO Buffer Payload Storage Size Select<br>b6 b5 b4<br>0 0 0: 8 bytes<br>0 0 1: 12 bytes<br>0 1 0: 16 bytes<br>0 1 1: 20 bytes<br>1 0 0: 24 bytes<br>1 0 1: 32 bytes<br>1 1 0: 48 bytes<br>1 1 1: 64 bytes |
| 3, 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | RFIE | Receive FIFO Interrupt Enable<br>0: Receive FIFO interrupt is disabled.<br>1: Receive FIFO interrupt is enabled. |
| 0 | RFE | Receive FIFO Buffer Enable<br>0: No receive FIFO buffer is used.<br>1: Receive FIFO buffers are used. |

### RFIGCV[2:0] Bits

These bits are used to specify the number of received messages for generating a receive FIFO interrupt request when the RFIM bit is set to 0 with a fraction for the total number of buffers (the setting of RFDC[2:0]).

When the RFDC[2:0] bits are set to $001_B$ (4 messages), set the RFIGCV[2:0] bits to $001_B$, $011_B$, $101_B$, or $111_B$. Modify these bits only in global reset mode.

### RFIM Bit

This bit is used to select a FIFO interrupt source. Modify this bit only in global reset mode.

### RFDC[2:0] Bits

These bits are used to select the number of messages that can be stored in a single receive FIFO buffer. When these bits are set to $000_B$, no receive FIFO buffer should be used. Modify these bits only in global reset mode.

### RFPLS[2:0] Bits

These bits are used to select the maximum payload size that can be stored in the receive FIFO buffer. Modify these bits only in global reset mode.

### RFIE Bit

Setting the RFIE bit to 1 enables receive FIFO interrupts. Modify this bit when the RFE bit is set to 0 (no receive FIFO buffer is used).

### RFE Bit

Setting the RFE bit to 1 makes receive FIFO buffers available. Clearing this bit to 0 sets the RFEMP flag in the RSCFDnCFDRFSTSx register to 1 (buffer empty). Modify this bit in global operating mode or global test mode. Set this bit to 1 with another instruction after the settings to all bits in the RSCFDnCFDRFCCx register have been done. This bit is cleared to 0 in global reset mode.

### 21.5.7.2 RSCFDnCFDRFSTSx — Receive FIFO Buffer Status Register (x = 0 to 7)

**Access:** RSCFDnCFDRFSTSx register can be read or written in 32-bit units
RSCFDnCFDRFSTSxL register can be read or written in 16-bit units
RSCFDnCFDRFSTSxLL register can be read or written in 8-bit units
RSCFDnCFDRFSTSxLH register is a read-only register that can be read in 8-bit units

**Address:** RSCFDnCFDRFSTSx: <RSCANn_base> + $00D8_H$ + ($04_H$ × x)

RSCFDnCFDRFSTSxL: <RSCANn_base> + $00D8_H$ + ($04_H$ × x)

RSCFDnCFDRFSTSxLL: <RSCANn_base> + $00D8_H$ + ($04_H$ × x),
RSCFDnCFDRFSTSxLH: <RSCANn_base> + $00D9_H$ + ($04_H$ × x)

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFMC[7:0] | | | | | | | | — | — | — | — | RFIF | RFMLT | RFFLL | RFEMP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W*1 | R/W*1 | R | R |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.129 RSCFDnCFDRFSTSx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 15 to 8 | RFMC[7:0] | Receive FIFO Unread Message Counter<br>The number of unread messages stored in the receive FIFO buffer is displayed. |
| 7 to 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | RFIF | Receive FIFO Interrupt Request Flag<br>0: No receive FIFO interrupt request is present.<br>1: A receive FIFO interrupt request is present. |
| 2 | RFMLT | Receive FIFO Message Lost Flag<br>0: No receive FIFO message is lost.<br>1: A receive FIFO message is lost. |
| 1 | RFFLL | Receive FIFO Buffer Full Status Flag<br>0: The receive FIFO buffer is not full.<br>1: The receive FIFO buffer is full. |
| 0 | RFEMP | Receive FIFO Buffer Empty Status Flag<br>0: The receive FIFO buffer contains unread message.<br>1: The receive FIFO buffer contains no unread message (buffer empty). |

#### RFMC[7:0] Flag

This flag indicates the number of unread messages in the receive FIFO buffer. This flag becomes $00_H$ when the RFE bit in the RSCFDnCFDRFCCx register is set to 0.

This flag is 00H in global reset mode.

### RFIF Flag

This flag is set to 1 when the receive FIFO interrupt request generation conditions set by the RFIGCV[2:0] bits and the RFIM bit in the RSCFDnCFDRFCCx register are met. This flag is cleared to 0 in global reset mode or by writing 0 to this flag. Modify this bit in global operating mode or global test mode.
To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared. When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

### RFMLT Flag

This flag is set to 1 when an attempt is made to store a new message while the receive FIFO buffer is full. In this case, the new message is discarded.

This flag is cleared to 0 in global reset mode or by writing 0 to this flag.

Modify this bit in global operating mode or global test mode.

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared. When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

### RFFLL Flag

This flag is set to 1 when the number of messages stored in the receive FIFO buffer matches the FIFO buffer depth set by the RFDC[2:0] bits in the RSCFDnCFDRFCCx register.

If the number of messages stored in the receive FIFO buffer becomes smaller than the FIFO buffer depth set by the RFDC[2:0] bits, this flag is cleared to 0. This flag is also cleared to 0 when the RFE bit in the RSCFDnCFDRFCCx register is set to 0 (no receive FIFO buffer is used) or in global reset mode.

### RFEMP Flag

This flag is set to 1 when all messages in the receive FIFO buffer have been read. This flag is also set to 1 when the RFE bit in the RSCFDnCFDRFCCx register is 0 or in global reset mode.

This flag is cleared to 0 when even a single received message has been stored in the receive FIFO buffer.

**NOTE**

To clear the RFMLT or RFIF flag to 0, use a store instruction to write “0” to the given flag and “1” to other flags.

### 21.5.7.3 RSCFDnCFDRFPCTRx — Receive FIFO Buffer Pointer Control Register (x = 0 to 7)

**Access:** RSCFDnCFDRFPCTRx register is a write-only register that can be written in 32-bit units
RSCFDnCFDRFPCTRxL register is a write-only register that can be written in 16-bit units
RSCFDnCFDRFPCTRxLL register is a write-only register that can be written in 8-bit units

**Address:** RSCFDnCFDRFPCTRx: <RSCANn_base> + $00F8_H$ + ($04_H$ × x)

RSCFDnCFDRFPCTRxL: <RSCANn_base> + $00F8_H$ + ($04_H$ × x)

RSCFDnCFDRFPCTRxLL: <RSCANn_base> + $00F8_H$ + ($04_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RFPC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**Table 21.130 RSCFDnCFDRFPCTRx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing, write the value after reset. |
| 7 to 0 | RFPC[7:0] | Receive FIFO Pointer Control<br>When these bits are set to $FF_H$, the read pointer moves to the next unread message in the receive FIFO buffer. |

When the RFDMAEx value in the RSCFDnCFDCDTCT register is 1 (DMA transfer request enabled), do not write a value to this register.

#### RFPC[7:0] Bits

When the RFPC[7:0] bits are set to $FF_H$, the read pointer moves to the next unread message in the receive FIFO buffer. At this time, the RFMC[7:0] (receive FIFO unread message counter) value in the RSCFDnCFDRFSTSx register is decremented. Read the RSCFDnCFDRFIDx, RSCFDnCFDRFPTRx, RSCFDnCFDRFFDSTSx, and RSCFDnCFDRFDFd_x registers to read messages in the receive FIFO buffer, and then write $FF_H$ to the RFPC[7:0] bits.

When writing $FF_H$ to these bits, make sure that the RFE bit in the RSCFDnCFDRFCCx register is set to 1 (receive FIFO buffers are used) and the RFEMP flag in the RSCFDnCFDRFSTSx register is 0 (the receive FIFO buffer contains unread messages).

### 21.5.7.4 RSCFDnCFDRFIDx — Receive FIFO Buffer Access ID Register (x = 0 to 7)

**Access:** RSCFDnCFDRFIDx register is a read-only register that can be read in 32-bit units
RSCFDnCFDRFIDxL, RSCFDnCFDRFIDxH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDRFIDxLL, RSCFDnCFDRFIDxLH, RSCFDnCFDRFIDxHL, RSCFDnCFDRFIDxHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDRFIDx: <RSCANn_base> + $3000_H$ + ($80_H$ × x)

RSCFDnCFDRFIDxL: <RSCANn_base> + $3000_H$ + ($80_H$ × x),
RSCFDnCFDRFIDxH: <RSCANn_base> + $3002_H$ + ($80_H$ × x)

RSCFDnCFDRFIDxLL: <RSCANn_base> + $3000_H$ + ($80_H$ × x),
RSCFDnCFDRFIDxLH: <RSCANn_base> + $3001_H$ + ($80_H$ × x),
RSCFDnCFDRFIDxHL: <RSCANn_base> + $3002_H$ + ($80_H$ × x),
RSCFDnCFDRFIDxHH: <RSCANn_base> + $3003_H$ + ($80_H$ × x)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFIDE | RFRTR | — | RFID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.131 RSCFDnCFDRFIDx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | RFIDE | Receive FIFO Buffer IDE<br>0: Standard ID<br>1: Extended ID |
| 30 | RFRTR | Receive FIFO Buffer RTR/RRS<br>• When the received message is a classical CAN frame<br>0: Data frame<br>1: Remote frame<br>• When the received message is a CAN FD frame<br>The RRS bit value of the received message can be read. |
| 29 | Reserved | When read, the value after reset is returned. |
| 28 to 0 | RFID[28:0] | Receive FIFO Buffer ID Data<br>The standard ID or extended ID of received message can be read.<br>Read bits b10 to b0 for standard ID. Bits b28 to b11 are read as 0. |

#### RFIDE Bit

This bit indicates the ID format (standard ID or extended ID) of the message stored in the receive FIFO buffer.

#### RFRTR Bit

When the received message is a classical CAN frame, this bit indicates the frame format (data frame or remote frame) of the message stored in the receive FIFO buffer. When the received message is a CAN FD frame, this bit indicates the RRS bit value in the message.

#### RFID[28:0] Bits

These bits indicate the ID of the message stored in the receive FIFO buffer.

### 21.5.7.5 RSCFDnCFDRFPTRx — Receive FIFO Buffer Access Pointer Register (x = 0 to 7)

**Access:** RSCFDnCFDRFPTRx register is a read-only register that can be read in 32-bit units
RSCFDnCFDRFPTRxL, RSCFDnCFDRFPTRxH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDRFPTRxLL, RSCFDnCFDRFPTRxLH, RSCFDnCFDRFPTRxHL, RSCFDnCFDRFPTRxHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDRFPTRx: <RSCANn_base> + $3004_H$ + ($80_H$ × x)

RSCFDnCFDRFPTRxL: <RSCANn_base> + $3004_H$ + ($80_H$ × x),
RSCFDnCFDRFPTRxH: <RSCANn_base> + $3006_H$ + ($80_H$ × x)

RSCFDnCFDRFPTRxLL: <RSCANn_base> + $3004_H$ + ($80_H$ × x),
RSCFDnCFDRFPTRxLH: <RSCANn_base> + $3005_H$ + ($80_H$ × x),
RSCFDnCFDRFPTRxHL: <RSCANn_base> + $3006_H$ + ($80_H$ × x),
RSCFDnCFDRFPTRxHH: <RSCANn_base> + $3007_H$ + ($80_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFDLC[3:0] | | | | RFPTR[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFTS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.132 RSCFDnCFDRFPTRx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | RFDLC[3:0] | Receive FIFO Buffer DLC Data |

| b31 | b30 | b29 | b28 | Classical CAN Frame | CAN FD Frame |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 data bytes | |
| 0 | 0 | 0 | 1 | 1 data byte | |
| 0 | 0 | 1 | 0 | 2 data bytes | |
| 0 | 0 | 1 | 1 | 3 data bytes | |
| 0 | 1 | 0 | 0 | 4 data bytes | |
| 0 | 1 | 0 | 1 | 5 data bytes | |
| 0 | 1 | 1 | 0 | 6 data bytes | |
| 0 | 1 | 1 | 1 | 7 data bytes | |
| 1 | 0 | 0 | 0 | 8 data bytes | |
| 1 | 0 | 0 | 1 | 8 data bytes | 12 data bytes |
| 1 | 0 | 1 | 0 | | 16 data bytes |
| 1 | 0 | 1 | 1 | | 20 data bytes |
| 1 | 1 | 0 | 0 | | 24 data bytes |
| 1 | 1 | 0 | 1 | | 32 data bytes |
| 1 | 1 | 1 | 0 | | 48 data bytes |
| 1 | 1 | 1 | 1 | | 64 data bytes |

| | | |
|---|---|---|
| 27 to 16 | RFPTR[11:0] | Receive FIFO Buffer Label Data<br>Label information of the received message can be read. |
| 15 to 0 | RFTS[15:0] | Receive FIFO Buffer Timestamp Data<br>Timestamp value of the received message can be read. |

#### RFDLC[3:0] Bits

These bits indicate the data length of the message stored in the receive FIFO buffer.

#### RFPTR[11:0] Bits

These bits indicate the label information of the message stored in the receive FIFO buffer.

#### RFTS[15:0] Bits

These bits indicate the timestamp value of the message stored in the receive FIFO buffer.

### 21.5.7.6 RSCFDnCFDRFFDSTSx — Receive FIFO CAN FD Status Register (x = 0 to 7)

**Access:** RSCFDnCFDRFFDSTSx register is a read-only register that can be read in 32-bit units
RSCFDnCFDRFFDSTSxL register is a read-only register that can be read in 16-bit units
RSCFDnCFDRFFDSTSxLL register is a read-only register that can be read in 8-bit units

**Address:** RSCFDnCFDRFFDSTSx: <RSCANn_base> + $3008_H$ + ($80_H$ × x)

RSCFDnCFDRFFDSTSxL: <RSCANn_base> + $3008_H$ + ($80_H$ × x)

RSCFDnCFDRFFDSTSxLL: <RSCANn_base> + $3008_H$ + ($80_H$ × x)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | RFFDF | RFBRS | RFESI |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.133 RSCFDnCFDRFFDSTSx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 | RFFDF | FDF<br>0: Classical CAN frame<br>1: CAN FD frame |
| 1 | RFBRS | BRS<br>0: The bit rate in the data area does not change.<br>1: The bit rate in the data area changes. |
| 0 | RFESI | ESI<br>0: Error active node<br>1: Error passive node |

#### RFFDF Bit

This bit indicates the FD format (classical CAN frame or CAN FD frame) of the message stored in the receive FIFO buffer.

#### RFBRS Bit

When the RFFDF bit is 1, this bit indicates the BRS bit value of the message stored in the receive FIFO buffer. When the RFFDF bit is 0, this bit is always read as 0.

#### RFESI Bit

When the RFFDF bit is 1, this bit indicates the ESI bit value of the message stored in the receive FIFO buffer. When the RFFDF bit is 0, this bit is always read as 0.

### 21.5.7.7 RSCFDnCFDRFDFd_x — Receive FIFO Buffer Access Data Field d Register (d = 0 to 15, x = 0 to 7)

**Access:** RSCFDnCFDRFDFd_x register is a read-only register that can be read in 32-bit units
RSCFDnCFDRFDFd_xL, RSCFDnCFDRFDFd_xH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDRFDFd_xLL, RSCFDnCFDRFDFd_xLH, RSCFDnCFDRFDFd_xHL, RSCFDnCFDRFDFd_xHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDRFDFd_x: <RSCANn_base> + $300C_H$ + ($04_H$ × d) + ($80_H$ × x)

RSCFDnCFDRFDFd_xL: <RSCANn_base> + $300C_H$ + ($04_H$ × d) + ($80_H$ × x),
RSCFDnCFDRFDFd_xH: <RSCANn_base> + $300E_H$ + ($04_H$ × d) + ($80_H$ × x)

RSCFDnCFDRFDFd_xLL: <RSCANn_base> + $300C_H$ + ($04_H$ × d) + ($80_H$ × x),
RSCFDnCFDRFDFd_xLH: <RSCANn_base> + $300D_H$ + ($04_H$ × d) + ($80_H$ × x),
RSCFDnCFDRFDFd_xHL: <RSCANn_base> + $300E_H$ + ($04_H$ × d) + ($80_H$ × x),
RSCFDnCFDRFDFd_xHH: <RSCANn_base> + $300F_H$ + ($04_H$ × d) + ($80_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFDB4 × d + 3 [7:0] | | | | | | | | RFDB4 × d + 2 [7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFDB4 × d + 1 [7:0] | | | | | | | | RFDB4 × d + 0 [7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.134 RSCFDnCFDRFDFd_x Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RFDB4 × d + 3 [7:0] | Receive Buffer Data Byte 4 × d + 3<br>Receive Buffer Data Byte 4 × d + 2<br>Receive Buffer Data Byte 4 × d + 1<br>Receive Buffer Data Byte 4 × d + 0 |
| 23 to 16 | RFDB4 × d + 2 [7:0] | |
| 15 to 8 | RFDB4 × d + 1 [7:0] | Data for a message stored in the receive buffer can be read. |
| 7 to 0 | RFDB4 × d + 0 [7:0] | |

When the RFDLC[3:0] value in the RSCFDnCFDRFPTRx register is smaller than the payload storage size of the receive FIFO buffer, data bytes for which no data is set are read as $00_H$.

Specify the payload storage size of the receive FIFO buffer by the RFPLS[2:0] bits in the RSCFDnCFDRFCCx register. Do not read or write the RSCFDnCFDRFDFd_x register corresponding to an area larger than the specified size.

## 21.5.8 Transmit/Receive FIFO Buffer Related Registers

### 21.5.8.1 RSCFDnCFDCFCCk — Transmit/receive FIFO Buffer Configuration and Control Register k (k = 0 to 17)

**Access:** RSCFDnCFDCFCCk register can be read or written in 32-bit units
RSCFDnCFDCFCCkL, RSCFDnCFDCFCCkH registers can be read or written in 16-bit units
RSCFDnCFDCFCCkLL, RSCFDnCFDCFCCkLH, RSCFDnCFDCFCCkHL, RSCFDnCFDCFCCkHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDCFCCk: <RSCANn_base> + $0118_H$ + ($04_H$ × k)

RSCFDnCFDCFCCkL: <RSCANn_base> + $0118_H$ + ($04_H$ × k),
RSCFDnCFDCFCCkH: <RSCANn_base> + $011A_H$ + ($04_H$ × k)

RSCFDnCFDCFCCkLL: <RSCANn_base> + $0118_H$ + ($04_H$ × k),
RSCFDnCFDCFCCkLH: <RSCANn_base> + $0119_H$ + ($04_H$ × k),
RSCFDnCFDCFCCkHL: <RSCANn_base> + $011A_H$ + ($04_H$ × k),
RSCFDnCFDCFCCkHH: <RSCANn_base> + $011B_H$ + ($04_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFITT[7:0] | | | | | | | | CFTML[3:0] | | | | CFITR | CFITSS | CFM[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFIGCV[2:0] | | | CFIM | — | CFDC[2:0] | | | — | CFPLS[2:0] | | | — | CFTXIE | CFRXIE | CFE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

**Table 21.135 RSCFDnCFDCFCCk Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | CFITT[7:0] | Set a message transmission interval.<br>Set Value: $00_H$ to $FF_H$ |
| 23 to 20 | CFTML[3:0] | Transmit Buffer Link Configuration<br>Set the transmit buffer number to be linked to the transmit/receive FIFO buffer. |
| 19 | CFITR | Transmit/Receive FIFO Interval Timer Resolution<br>0: Clock dividing pclk/2 by (ITRCP[15:0] bits)<br>1: Clock dividing pclk/2 by (ITRCP[15:0] bits × 10) |
| 18 | CFITSS | Transmit/Receive FIFO Interval Timer Clock Source Select<br>0: Interval timer clock source selected by the CFITR bit<br>1: Interval timer clock source is the nominal bit time clock for the channel to which the FIFO is linked. |
| 17, 16 | CFM[1:0] | Transmit/Receive FIFO Mode Select<br>b17 b16<br>0 0: Receive mode<br>0 1: Transmit mode<br>1 0: Gateway mode<br>1 1: Setting prohibited |
| 15 to 13 | CFIGCV[2:0] | Transmit/Receive FIFO Receive Interrupt Request Timing Select<br>b15 b14 b13<br>0 0 0: When FIFO is 1/8 full.<br>0 0 1: When FIFO is 2/8 full.<br>0 1 0: When FIFO is 3/8 full.<br>0 1 1: When FIFO is 4/8 full.<br>1 0 0: When FIFO is 5/8 full.<br>1 0 1: When FIFO is 6/8 full.<br>1 1 0: When FIFO is 7/8 full.<br>1 1 1: When FIFO is full. |

Table 21.135 RSCFDnCFDCFCCk Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 12 | CFIM | Transmit/Receive FIFO Interrupt Source Select<br>0:<br>• Receive mode/gateway mode<br>When the number of received messages has met the condition set by the CFIGCV[2:0] bits, a FIFO receive interrupt request is generated.<br>• Transmit mode/gateway mode<br>When the buffer becomes empty upon completion of message transmission, a FIFO transmit interrupt request is generated.<br>1:<br>• Receive mode/gateway mode<br>A FIFO receive interrupt request is generated each time a message has been received.<br>• Transmit mode/gateway mode<br>A FIFO transmit interrupt request is generated each time a message has been transmitted. |
| 11 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 10 to 8 | CFDC[2:0] | Transmit/Receive FIFO Buffer Depth Configuration<br>b10 b9 b8<br>0 0 0: 0 messages<br>0 0 1: 4 messages<br>0 1 0: 8 messages<br>0 1 1: 16 messages<br>1 0 0: 32 messages<br>1 0 1: 48 messages<br>1 1 0: 64 messages<br>1 1 1: 128 messages |
| 7 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 6 to 4 | CFPLS[2:0] | Transmit/Receive FIFO Buffer Payload Storage Size Select<br>b6 b5 b4<br>0 0 0: 8 bytes<br>0 0 1: 12 bytes<br>0 1 0: 16 bytes<br>0 1 1: 20 bytes<br>1 0 0: 24 bytes<br>1 0 1: 32 bytes<br>1 1 0: 48 bytes<br>1 1 1: 64 bytes |
| 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 | CFTXIE | Transmit/Receive FIFO Transmit Interrupt Enable<br>0: Transmit/receive FIFO transmit interrupt is disabled.<br>1: Transmit/receive FIFO transmit interrupt is enabled. |
| 1 | CFRXIE | Transmit/Receive FIFO Receive Interrupt Enable<br>0: Transmit/receive FIFO receive interrupt is disabled.<br>1: Transmit/receive FIFO receive interrupt is enabled. |
| 0 | CFE | Transmit/Receive FIFO Buffer Enable<br>0: No transmit/receive FIFO buffer is used.<br>1: Transmit/receive FIFO buffers are used. |

### CFITT[7:0] Bits

These bits are used to set a message transmission interval when transmitting messages continuously from a transmit/receive FIFO buffer whose CFM[1:0] bits are set to $01_B$ (transmit mode) or $10_B$ (gateway mode).

Clear the CFE bit to 0 (no transmit/receive FIFO buffer is used) before modifying the CFITT[7:0] bits.

#### CFTML[3:0] Bits

These bits are used to set the number of transmit buffer on the channel which will be linked to transmit/receive FIFO buffer k when the CFM[1:0] bits are set to $01_B$ (transmit mode) or $10_B$ (gateway mode). There are three transmit/receive FIFO buffers per channel, so channel number m of FIFO buffer k is calculated as m = k/3 (integer division). The actual assigned transmit buffer number p linked to FIFO buffer k will be ((16 × m) + CFTML[3:0]).

See **Table 21.91** and **Table 21.92**, for the relationship between transmit/receive FIFO buffer k and transmit buffer p.

Setting the CFDC[2:0] bits to $001_B$ or more enables the setting of the CFTML[3:0] bits.

Do not link to any transmit buffer which is already allocated to a transmit queue on the identical channel or to another transmit/receive FIFO buffer. Modify these bits only in global reset mode.

#### CFITR Bit

This bit is enabled when the CFITSS bit is 0.

When this bit is 0, the interval timer clock source is the pclk/2 clock divided by the value of the ITRCP[15:0] bits in the RSCFDnCFDGCFG register.

When this bit is 1, the interval timer clock source is the pclk/2 clock divided by the value of the ITRCP[15:0] bits in the RSCFDnCFDGCFG register × 10.

Modify this bit while the CFE bit is set to 0 (no transmit/receive FIFO buffer is used).

#### CFITSS Bit

When this bit is 0, the clock selected by the CFITR bit is the count source of the interval timer.

When this bit is 1, the nominal bit time clock of the channel to which the FIFO is linked is the count source of the interval timer. Use this count source only for the channel does not handle the CAN FD frames.

Modify this bit while the CFE bit is set to 0 (no transmit/receive FIFO buffer is used).

#### CFM[1:0] Bits

These bits are used to select transmit/receive FIFO mode. Modify these bits only in global reset mode.

#### CFIGCV[2:0] Bits

These bits are used to specify the number of received messages for generating a transmit/receive FIFO receive interrupt request when the CFM[1:0] bits are set to $00_B$ (receive mode) or $10_B$ (gateway mode) and the CFIM bit is set to 0 with a fraction for the total number of buffers (the setting of CFDC[2:0]).

When the CFDC[2:0] bits are set to $001_B$ (4 messages), set the CFIGCV[2:0] bits to $001_B$, $011_B$, $101_B$, or $111_B$.

Modify these bits only in global reset mode.

#### CFIM Bit

This bit is used to select a transmit/receive FIFO interrupt source. Modify this bit only in global reset mode.

#### CFDC[2:0] Bits

These bits are used to set the number of messages that can be stored in a single transmit/receive FIFO buffer. When these bits are set to $000_B$, do not use a transmit/receive FIFO buffer. Modify these bits only in global reset mode.

#### CFPLS[2:0] Bits

These bits are used to select the maximum payload size that can be stored in the transmit/receive FIFO buffer. Modify these bits only in global reset mode.

#### CFTXIE Bit

When this bit is set to 1 and the CFTXIF flag in the RSCFDnCFDCFSTSk register is set to 1, a transmit/receive FIFO transmit interrupt request is generated.

Modify this bit with the CFE bit set to 0 (no transmit/receive FIFO buffer is used).

#### CFRXIE Bit

When this bit is set to 1 and the CFRXIF flag in the RSCFDnCFDCFSTSk register is set to 1, a transmit/receive FIFO receive interrupt request is generated.

Modify this bit with the CFE bit set to 0.

#### CFE Bit

Setting this bit to 1 makes transmit/receive FIFO buffers available.

When this bit is set to 0 in transmit mode or gateway mode, if a message in the transmit/receive FIFO buffer is being transmitted or will be transmitted next, the transmit/receive FIFO buffer becomes empty after completion of transmission of that message, or upon detection of a CAN bus error, or arbitration-lost. In other cases or in receive mode, the transmit/receive FIFO buffer becomes empty immediately.

This bit is cleared to 0 when the following conditions are met.

- Receive mode: Global reset mode
- Transmit mode or gateway mode: Channel reset mode

Modify this bit in the following mode.

- Receive mode: Global operating mode or global test mode
- Transmit mode or gateway mode: Channel communication mode or channel halt mode

After all other bits in the RSCFDnCFDCFCCk register have been set, set this bit to 1 by using other instructions.

### 21.5.8.2 RSCFDnCFDCFSTSk — Transmit/receive FIFO Buffer Status Register (k = 0 to 17)

**Access:** RSCFDnCFDCFSTSk register can be read or written in 32-bit units
RSCFDnCFDCFSTSkL register can be read or written in 16-bit units
RSCFDnCFDCFSTSkLL register can be read or written in 8-bit units
RSCFDnCFDCFSTSkLH register is a read-only register that can be read in 8-bit units

**Address:** RSCFDnCFDCFSTSk: <RSCANn_base> + $0178_H$ + ($04_H$ × k)

RSCFDnCFDCFSTSkL: <RSCANn_base> + $0178_H$ + ($04_H$ × k)

RSCFDnCFDCFSTSkLL: <RSCANn_base> + $0178_H$ + ($04_H$ × k),
RSCFDnCFDCFSTSkLH: <RSCANn_base> + $0179_H$ + ($04_H$ × k)

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFMC[7:0] | | | | | | | | — | — | — | CFTXIF | CFRXIF | CFMLT | CFFLL | CFEMP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W*1 | R/W*1 | R/W*1 | R | R |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.136 RSCFDnCFDCFSTSk Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 15 to 8 | CFMC[7:0] | Transmit/Receive FIFO Message Counter<br>The number of messages stored in the transmit/receive FIFO buffer. |
| 7 to 5 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 4 | CFTXIF | Transmit/Receive FIFO Transmit Interrupt Request Flag<br>0: No transmit/receive FIFO transmit interrupt request is present.<br>1: A transmit/receive FIFO transmit interrupt request is present. |
| 3 | CFRXIF | Transmit/Receive FIFO Receive Interrupt Request Flag<br>0: No transmit/receive FIFO receive interrupt request is present.<br>1: A transmit/receive FIFO receive interrupt request is present. |
| 2 | CFMLT | Transmit/Receive FIFO Message Lost Flag<br>0: No transmit/receive FIFO message is lost.<br>1: A transmit/receive FIFO message is lost. |
| 1 | CFFLL | Transmit/Receive FIFO Buffer Full Status Flag<br>0: The transmit/receive FIFO buffer is not full.<br>1: The transmit/receive FIFO buffer is full. |
| 0 | CFEMP | Transmit/Receive FIFO Buffer Empty Status Flag<br>0: The transmit/receive FIFO buffer contains messages.<br>1: The transmit/receive FIFO buffer contains no message (buffer empty). |

#### CFMC[7:0] Bits

The CFMC[7:0] bits indicate the following values that depend on the setting of the CFM[1:0] bits in the RSCFDnCFDCFCCk register.

- When CFM[1:0] value is $01_B$ (transmit mode): Number of untransmitted messages in the buffer
- When CFM[1:0] value is $00_B$ (receive mode): Number of unread received messages in the buffer
- When CFM[1:0] value is $10_B$ (gateway mode): Number of untransmitted received messages in the buffer

These bits are cleared to 0 when any of the following conditions is met.

- When CFM[1:0] value is $00_B$: In global reset mode
- When CFM[1:0] value is $01_B$ or $10_B$: In channel reset mode
- When the CFE bit in the RSCANnCFCCk register is cleared to 0.

#### CFTXIF Flag

The CFTXIF flag is set to 1 when any of the following conditions is met.

- When the CFM[1:0] bits are set to $01_B$ or $10_B$, and the factor selected by the CFIM bit in the RSCFDnCFDCFCCk register occurs

The CFTXIF flag is cleared to 0 when any of the following conditions is met.

- When 0 is written to the CFTXIF flag
- When the CFM[1:0] bits are set to $00_B$: In global reset mode
- When the CFM[1:0] bits are set to $01_B$ or $10_B$: In channel reset mode

Write 0 to this flag in global operating mode or global test mode.

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared. When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

#### CFRXIF Flag

The CFRXIF flag is set to 1 when any of the following conditions is met.

- When the CFM[1:0] bits are set to $00_B$ or $10_B$, and the factor selected by the CFIM bit in the RSCFDnCFDCFCCk register occurs

The CFRXIF flag is cleared to 0 when any of the following conditions is met.

- When 0 is written to the CFRXIF flag
- When the CFM[1:0] bits are set to $00_B$: In global reset mode
- When the CFM[1:0] bits are set to $01_B$ or $10_B$: In channel reset mode

Write 0 to this flag in global operating mode or global test mode.

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared.

When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

### CFMLT Flag

The CFMLT flag is set to 1 when any of the following conditions is met.

- When an attempt is made to store a new message while the transmit/receive FIFO buffer is full. In this case, the new message is discarded.

The CFMLT flag is cleared to 0 when any of the following conditions is met.

- When 0 is written to the CFMLT flag
- When the CFM[1:0] bits are set to $00_B$: In global reset mode
- When the CFM[1:0] bits are set to $01_B$ or $10_B$: In channel reset mode

Write 0 to this flag in global operating mode or global test mode

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared. When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

### CFFLL Flag

The CFFLL flag is set to 1 when any of the following conditions is met.

- When the number of messages stored in the transmit/receive FIFO buffer matches the FIFO buffer depth set by the CFDC[2:0] bits in the RSCFDnCFDCFCCk register.

The CFFLL flag is cleared to 0 when any of the following conditions is met.

- When the number of messages stored in the transmit/receive FIFO buffer becomes smaller than the FIFO buffer depth set by the CFDC[2:0] bits.
- When the CFE bit in the RSCFDnCFDCFCCk register is 0 (no transmit/receive FIFO buffer is used): When not performing transmit abort
- When the CFM[1:0] bits are set to $00_B$: In global reset mode
- When the CFM[1:0] bits are set to $01_B$ or $10_B$: In channel reset mode

### CFEMP Flag

The CFEMP flag is set to 1 when any of the following conditions is met.

- When the CFM[1:0] bits are set to $00_B$: All messages have been read, or in global reset mode
- When the CFM[1:0] bits are set to $01_B$ or $10_B$: All messages have been transmitted, or in channel reset mode
- When the CFE bit is 0 (no transmit/receive FIFO buffer is used): When not performing transmit abort

The CFEMP flag is cleared to 0 when any of the following conditions is met.

- When the CFM[1:0] bits are set to $00_B$ or $10_B$: When at least one received message has been stored in the transmit/receive FIFO buffer.
- When the CFM[1:0] bits are set to $01_B$: When a value of $FF_H$ has been written to the RSCFDnCFDCFPCTRk register after data was written to the RSCFDnCFDCFIDk, RSCFDnCFDCFPTRk, RSCFDnCFDCFFDCSTSk, and RSCFDnCFDCFDFd_k registers.

**NOTE**

To clear CFTXIF, CFRXIF, or CFMLT flag to 0, the program must write 0. When writing, use a store instruction, write “0” to the given flag and “1” to other flags.

### 21.5.8.3 RSCFDnCFDCFPCTRk — Transmit/receive FIFO Buffer Pointer Control Register (k = 0 to 17)

**Access:** RSCFDnCFDCFPCTRk register is a write-only register that can be written in 32-bit units
RSCFDnCFDCFPCTRkL register is a write-only register that can be written in 16-bit units
RSCFDnCFDCFPCTRkLL register is a write-only register that can be written in 8-bit units

**Address:** RSCFDnCFDCFPCTRk: <RSCANn_base> + $01D8_H$ + ($04_H$ × k)

RSCFDnCFDCFPCTRkL: <RSCANn_base> + $01D8_H$ + ($04_H$ × k)

RSCFDnCFDCFPCTRkLL: <RSCANn_base> + $01D8_H$ + ($04_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CFPC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**Table 21.137 RSCFDnCFDCFPCTRk Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing, write the value after reset. |
| 7 to 0 | CFPC[7:0] | Transmit/Receive FIFO Pointer Control<br>• Receive mode:<br>Writing $FF_H$ to these bits moves the read pointer to the next unread message in the transmit/receive FIFO buffer.<br>• Transmit mode:<br>Writing $FF_H$ to these bits moves the write pointer to the next stage of the transmit/receive FIFO buffer.<br>• Gateway mode:<br>Setting prohibited |

When the corresponding transmit/receive FIFO buffer is the first transmit/receive FIFO buffer (k = 3 × m) allocated to channel m and when the CFDMAEm bit in the RSCFDnCFDCDTCT register is 1 (DMA transfer request enabled), do not write a value to this register.

#### CFPC[7:0] Bits

- Receive mode (CFM[1:0] value in the RSCFDnCFDCFCCk register is $00_B$):
  Writing $FF_H$ to the CFPC[7:0] bits moves the read pointer to the next unread message in the transmit/receive FIFO buffer. At this time, the CFMC[7:0] value (transmit/receive FIFO message counter) in the RSCFDnCFDCFSTSk register is decremented. Read the RSCFDnCFDCFIDk, RSCFDnCFDCFPTRk, RSCFDnCFDCFFDCSTSk, and RSCFDnCFDCFDFd_k registers to read messages from the transmit/receive FIFO buffer, and then write $FF_H$ to the CFPC[7:0] bits.
  When writing $FF_H$ to these bits, make sure that the CFE bit in the RSCFDnCFDCFCCk register is set to 1 (transmit/receive FIFO buffers are used) and the CFEMP flag in the RSCFDnCFDCFSTSk register is 0 (the transmit/receive FIFO buffer contains messages).

- Transmit mode (CFM[1:0] value in the RSCFDnCFDCFCCk register is $01_B$):
  Writing $FF_H$ to the CFPC[7:0] bits stores the data written to the RSCFDnCFDCFIDk, RSCFDnCFDCFPTRk, RSCFDnCFDCFFDCSTSk, and RSCFDnCFDCFDFd_k registers in the transmit/receive FIFO buffer and moves the write pointer to the next stage of the transmit/receive FIFO buffer. At this time, the CFMC[7:0] value is incremented. Write transmit messages to the RSCFDnCFDCFIDk, RSCFDnCFDCFPTRk, RSCFDnCFDCFFDCSTSk, and RSCFDnCFDCFDFd_k registers before writing $FF_H$ to the CFPC[7:0] bits.
  When writing $FF_H$ to these bits, make sure that the CFE bit in the RSCFDnCFDCFCCk register is set to 1 and the CFFLL flag in the RSCFDnCFDCFSTSk register is 0 (the transmit/receive FIFO buffer is not full).
- Gateway mode (CFM[1:0] value in the RSCFDnCFDCFCCk register is $10_B$):
  Setting prohibited

### 21.5.8.4 RSCFDnCFDCFIDk — Transmit/receive FIFO Buffer Access ID Register (k = 0 to 17)

**Access:** RSCFDnCFDCFIDk register can be read or written in 32-bit units
RSCFDnCFDCFIDkL, RSCFDnCFDCFIDkH registers can be read or written in 16-bit units
RSCFDnCFDCFIDkLL, RSCFDnCFDCFIDkLH, RSCFDnCFDCFIDkHL, RSCFDnCFDCFIDkHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDCFIDk: <RSCANn_base> + $3400_H$ + ($80_H$ × k)

RSCFDnCFDCFIDkL: <RSCANn_base> + $3400_H$ + ($80_H$ × k),
RSCFDnCFDCFIDkH: <RSCANn_base> + $3402_H$ + ($80_H$ × k)

RSCFDnCFDCFIDkLL: <RSCANn_base> + $3400_H$ + ($80_H$ × k),
RSCFDnCFDCFIDkLH: <RSCANn_base> + $3401_H$ + ($80_H$ × k),
RSCFDnCFDCFIDkHL: <RSCANn_base> + $3402_H$ + ($80_H$ × k),
RSCFDnCFDCFIDkHH: <RSCANn_base> + $3403_H$ + ($80_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFIDE | CFRTR | THLEN | CFID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.138 RSCFDnCFDCFIDk Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | CFIDE | Transmit/Receive FIFO Buffer IDE<br>0: Standard ID<br>1: Extended ID |
| 30 | CFRTR | Transmit/Receive FIFO Buffer RTR/RRS<br>• When the CFM[1:0] value is $01_B$ (transmit mode)<br>– When the transmit message is a classical CAN frame<br>0: Data frame<br>1: Remote frame<br>– When the transmit message is a CAN FD frame<br>Write 0 to this bit.<br>• When the CFM[1:0] value is $00_B$ (receive mode)<br>– When the received message is a classical CAN frame<br>0: Data frame<br>1: Remote frame<br>– When the received message is a CAN FD frame<br>The RRS bit value of the received message can be read. |
| 29 | THLEN | Transmit History Data Store Enable<br>This bit is valid only when the CFM[1:0] value is $01_B$ (transmit mode).<br>0: Transmit history data is not stored in the buffer.<br>1: Transmit history data is stored in the buffer. |
| 28 to 0 | CFID[28:0] | Transmit/Receive FIFO Buffer ID Data<br>• When CFM[1:0] value is $01_B$ (transmit mode):<br>Set standard ID or extended ID. For standard ID, write an ID to bits 10 to 0 and write 0 to bits 28 to 11.<br>• When CFM[1:0] value is $00_B$ (receive mode):<br>Standard ID or extended ID in the received message can be read. For standard ID, read bits 10 to 0. Bits 28 to 11 are read as 0. |

This register is writable only when the CFM[1:0] value in the RSCFDnCFDCFCCk register is $01_B$ (transmit mode). This register is readable only when the CFM[1:0] value is $00_B$ (receive mode). This RSCFDnCFDCFIDk register should not be read or written when the CFM[1:0] value is $10_B$ (gateway mode).

#### CFIDE Bit

This bit indicates the ID format (standard ID or extended ID) of the received message stored in the transmit/receive FIFO buffer when the CFM[1:0] value is $00_B$. When the CFM[1:0] value is $01_B$, this bit is used to set the ID format of the message to be transmitted from the transmit/receive FIFO buffer.

#### CFRTR Bit

This bit indicates the data format (data frame or remote frame) of the received message stored in the transmit/receive FIFO buffer when the CFM[1:0] value is $00_B$. If the received message is a CAN FD frame, this bit indicates the RRS bit value of the received message.

When the CFM[1:0] value is $01_B$, this bit is used to set the data format of the message to be transmitted from the transmit/receive FIFO buffer.

When the CFFDF bit in the RSCFDnCFDCFFDCSTSk register is 1 (CAN FD frame), set this bit to 0.

#### THLEN Bit

When this bit is set to 1, the transmit history data (label information, buffer number, buffer type, and timestamp) of transmit messages is stored in the transmit history buffer after transmission is completed.

This bit is enabled when the CFM[1:0] value is $01_B$ (transmit mode).

#### CFID[28:0] Bits

These bits contain the ID of the received message stored in the transmit/receive FIFO buffer when the CFM[1:0] value is $00_B$.

When the CFM[1:0] value is $01_B$, these bits are used to set the ID of the message to be transmitted from the transmit/receive FIFO buffer.

### 21.5.8.5 RSCFDnCFDCFPTRk — Transmit/receive FIFO Buffer Access Pointer Register (k = 0 to 17)

**Access:** RSCFDnCFDCFPTRk register can be read or written in 32-bit units
RSCFDnCFDCFPTRkL, RSCFDnCFDCFPTRkH registers can be read or written in 16-bit units
RSCFDnCFDCFPTRkLL, RSCFDnCFDCFPTRkLH, RSCFDnCFDCFPTRkHL, RSCFDnCFDCFPTRkHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDCFPTRk: <RSCANn_base> + $3404_H$ + ($80_H$ × k)

RSCFDnCFDCFPTRkL: <RSCANn_base> + $3404_H$ + ($80_H$ × k),
RSCFDnCFDCFPTRkH: <RSCANn_base> + $3406_H$ + ($80_H$ × k)

RSCFDnCFDCFPTRkLL: <RSCANn_base> + $3404_H$ + ($80_H$ × k),
RSCFDnCFDCFPTRkLH: <RSCANn_base> + $3405_H$ + ($80_H$ × k),
RSCFDnCFDCFPTRkHL: <RSCANn_base> + $3406_H$ + ($80_H$ × k),
RSCFDnCFDCFPTRkHH: <RSCANn_base> + $3407_H$ + ($80_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFDLC[3:0] | | | | CFPTR[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFTS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.139 RSCFDnCFDCFPTRk Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | CFDLC[3:0] | Transmit/Receive FIFO Buffer DLC Data |

| b31 | b30 | b29 | b28 | Classical CAN Frame | CAN FD Frame |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 data bytes | |
| 0 | 0 | 0 | 1 | 1 data byte | |
| 0 | 0 | 1 | 0 | 2 data bytes | |
| 0 | 0 | 1 | 1 | 3 data bytes | |
| 0 | 1 | 0 | 0 | 4 data bytes | |
| 0 | 1 | 0 | 1 | 5 data bytes | |
| 0 | 1 | 1 | 0 | 6 data bytes | |
| 0 | 1 | 1 | 1 | 7 data bytes | |
| 1 | 0 | 0 | 0 | 8 data bytes | |
| 1 | 0 | 0 | 1 | 8 data bytes | 12 data bytes |
| 1 | 0 | 1 | 0 | | 16 data bytes |
| 1 | 0 | 1 | 1 | | 20 data bytes |
| 1 | 1 | 0 | 0 | | 24 data bytes |
| 1 | 1 | 0 | 1 | | 32 data bytes |
| 1 | 1 | 1 | 0 | | 48 data bytes |
| 1 | 1 | 1 | 1 | | 64 data bytes |

| | | |
|---|---|---|
| 27 to 16 | CFPTR[11:0] | Transmit/Receive FIFO Buffer Label Data<br>• When CFM[1:0] value is $01_B$ (transmit mode):<br>Set the label information to be stored in the transmit history buffer.<br>Only bits CFPTR[7:0] are valid.<br>• When CFM[1:0] value is $00_B$ (receive mode):<br>The label information of the received message can be read. |

Table 21.139 RSCFDnCFDCFPTRk Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | CFTS[15:0] | Transmit/Receive FIFO Buffer Timestamp Data<br>These bits are valid only when the CFM[1:0] value is $00_B$ (receive mode).<br>The timestamp value of the received message can be read. |

This register is writable only when the CFM[1:0] value in the RSCFDnCFDCFCCk register is $01_B$ (transmit mode). This register is readable only when the CFM[1:0] value is $00_B$ (receive mode). This register should not be read or written when the CFM[1:0] value is $10_B$ (gateway mode).

### CFDLC[3:0] Bits

These bits indicate the data length of the received message stored in the transmit/receive FIFO buffer when the CFM[1:0] value is $00_B$.

When the CFM[1:0] value is $01_B$, these bits are used to set the data length of the message to be transmitted from the transmit/receive FIFO buffer.

When the CFDLC[3:0] bits are set to $1001_B$ or more while the CFFDF bit in the RSCFDnCFDCFFDCSTSk register is 0 (CAN frame), 8-byte data is transmitted actually. When the CFFDF bit is 1 (CAN FD frame), the settable value range varies depending on the settings of the TMME bit in the RSCFDnCFDCmFDCFG register and the CFPLS[2:0] bits in the RSCFDnCFDCFCCk register.

- When TMME bit = 0 (transmit buffer merge mode disabled):
  A value of $0000_B$ to $1111_B$ is settable. If the specified data length exceeds the payload storage size specified by the CFPLS[2:0] bits, excessive payloads are padded by CCH.
- When TMME bit = 1 (transmit buffer merge mode enabled):
  Set the data length within the payload storage size specified by the CFPLS[2:0] bits.

### CFPTR[11:0] Bits

These bits indicate the label information attached to the received message stored in the transmit/receive FIFO buffer when the CFM[1:0] value is $00_B$. When the CFM[1:0] value is $01_B$, the CFPTR[7:0] value is stored in the transmit history buffer when message transmission has been completed.

### CFTS[15:0] Bits

These bits indicate the timestamp value of the message stored in the transmit/receive FIFO buffer.

These bits are valid when the CFM[1:0] value is $00_B$.

### 21.5.8.6 RSCFDnCFDCFFDCSTSk — Transmit/Receive FIFO CAN FD Configuration/Status Register (k = 0 to 17)

**Access:** RSCFDnCFDCFFDCSTSk register can be read or written in 32-bit units
RSCFDnCFDCFFDCSTSkL register can be read or written in 16-bit units
RSCFDnCFDCFFDCSTSkLL register can be read or written in 8-bit units

**Address:** RSCFDnCFDCFFDCSTSk: <RSCANn_base> + $3408_H$ + ($80_H$ × k)

RSCFDnCFDCFFDCSTSkL: <RSCANn_base> + $3408_H$ + ($80_H$ × k)

RSCFDnCFDCFFDCSTSkLL: <RSCANn_base> + $3408_H$ + ($80_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CFFDF | CFBRS | CFESI |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 21.140 RSCFDnCFDCFFDCSTSk Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 | CFFDF | FDF<br>0: Classical CAN frame<br>1: CAN FD frame |
| 1 | CFBRS | BRS<br>0: The bit rate in the data area does not change.<br>1: The bit rate in the data area changes. |
| 0 | CFESI | ESI<br>0: Error active node<br>1: Error passive node |

This register is writable only when the CFM[1:0] value in the RSCFDnCFDCFCCk register is $01_B$ (transmit mode). This register is readable only when the CFM[1:0] value is $00_B$ (receive mode). Do not read or write this register when the CFM[1:0] value is $10_B$ (gateway mode).

#### CFFDF Bit

When the CFM[1:0] value is $00_B$, this bit indicates the FD format (classical CAN frame or CAN FD frame) of the message stored in the transmit/receive FIFO buffer. When the CFM[1:0] value is $01_B$, this bit is used to set the FD format of the message to be transmitted from the transmit/receive FIFO buffer.

#### CFBRS Bit

When the CFM[1:0] value is $00_B$, if the CFFDF bit is 1, this bit indicates the BRS bit value of the message stored in the transmit/receive FIFO buffer. If the CFFDF bit is 0, this bit is always read as 0.

When the CFM[1:0] value is $01_B$, if the CFFDF bit is 1, this bit is used to set the BRS bit value of the message to be transmitted from the transmit/receive FIFO buffer. If the CFFDF bit is 0, write 0 to this bit.

**CFESI Bit**

When the CFM[1:0] value is $00_B$, if the CFFDF bit is 1, this bit indicates the ESI bit value of the message stored in the transmit/receive FIFO buffer. If the CFFDF bit is 0, this bit is always read as 0.

When the CFM[1:0] value is $01_B$, if the CFFDF bit is 1, this bit is used to set the ESI bit value of the message to be transmitted from the transmit/receive FIFO buffer. The set value is transmitted when the ESIC bit in the RSCFDnCFDCmFDCFG register is 1 and the channel is in the error active state. When the channel is in the error passive state, the ESI bit value that shows an error passive node is transmitted regardless of this bit value. When the CFFDF bit is 0, write 0 to this bit.

### 21.5.8.7 RSCFDnCFDCFDFd_k — Transmit/receive FIFO Buffer Access Data Field d Register (d = 0 to 15, k = 0 to 17)

**Access:** RSCFDnCFDCFDFd_k register can be read or written in 32-bit units
RSCFDnCFDCFDFd_kL, RSCFDnCFDCFDFd_kH registers can be read or written in 16-bit units
RSCFDnCFDCFDFd_kLL, RSCFDnCFDCFDFd_kLH, RSCFDnCFDCFDFd_kHL, RSCFDnCFDCFDFd_kHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDCFDFd_k: <RSCANn_base> + $340C_H$ + ($04_H$ × d) + ($80_H$ × k)

RSCFDnCFDCFDFd_kL: <RSCANn_base> + $340C_H$ + ($04_H$ × d) + ($80_H$ × k),
RSCFDnCFDCFDFd_kH: <RSCANn_base> + $340E_H$ + ($04_H$ × d) + ($80_H$ × k)

RSCFDnCFDCFDFd_kLL: <RSCANn_base> + $340C_H$ + ($04_H$ × d) + ($80_H$ × k),
RSCFDnCFDCFDFd_kLH: <RSCANn_base> + $340D_H$ + ($04_H$ × d) + ($80_H$ × k),
RSCFDnCFDCFDFd_kHL: <RSCANn_base> + $340E_H$ + ($04_H$ × d) + ($80_H$ × k),
RSCFDnCFDCFDFd_kHH: <RSCANn_base> + $340F_H$ + ($04_H$ × d) + ($80_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFDB4 × d + 3 [7:0] | | | | | | | | CFDB4 × d + 2 [7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFDB4 × d + 1 [7:0] | | | | | | | | CFDB4 × d + 0 [7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.141 RSCFDnCFDCFDFd_k Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | CFDB4 × d + 3 [7:0] | Transmit/Receive FIFO Buffer Data Byte 4 × d + 3<br>Transmit/Receive FIFO Buffer Data Byte 4 × d + 2<br>Transmit/Receive FIFO Buffer Data Byte 4 × d + 1<br>Transmit/Receive FIFO Buffer Data Byte 4 × d + 0<br>• When CFM[1:0] value is $01_B$ (transmit mode): Set the transmit/receive FIFO buffer data.<br>• When CFM[1:0] value is $00_B$ (receive mode): The message data stored in the transmit/receive FIFO buffer can be read. |
| 23 to 16 | CFDB4 × d + 2 [7:0] | |
| 15 to 8 | CFDB4 × d + 1 [7:0] | |
| 7 to 0 | CFDB4 × d + 0 [7:0] | |

This register is writable only when the CFM[1:0] value in the RSCFDnCFDCFCCk register is $01_B$ (transmit mode).

This register is readable only when the CFM[1:0] value is $00_B$ (receive mode). When the CFDLC[3:0] value in the RSCFDnCFDCFPTRk register is smaller than the payload storage size of the transmit/receive FIFO buffer, data bytes for which no data is set are read as $00_H$.

Specify the payload storage size of the transmit/receive FIFO buffer by the CFPLS[2:0] bits in the RSCFDnCFDCFCCk register. Do not read or write the RSCFDnCFDCFDFd_k register corresponding to an area larger than the specified size.

This register should not be read or written when the CFM[1:0] value is $10_B$ (gateway mode).

## 21.5.9 Details of FIFO Status-related Registers

### 21.5.9.1 RSCFDnCFDFESTS — FIFO Empty Status Register

**Access:** RSCFDnCFDFESTS register is a read-only register that can be read in 32-bit units
RSCFDnCFDFESTSL, RSCFDnCFDFESTSH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDFESTSLL, RSCFDnCFDFESTSLH, RSCFDnCFDFESTSHL, RSCFDnCFDFESTSHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDFESTS: <RSCANn_base> + $0238_H$

RSCFDnCFDFESTSL: <RSCANn_base> + $0238_H$,
RSCFDnCFDFESTSH: <RSCANn_base> + $023A_H$

RSCFDnCFDFESTSLL: <RSCANn_base> + $0238_H$,
RSCFDnCFDFESTSLH: <RSCANn_base> + $0239_H$,
RSCFDnCFDFESTSHL: <RSCANn_base> + $023A_H$,
RSCFDnCFDFESTSHH: <RSCANn_base> + $023B_H$

**Value after reset:** $03FF\ FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CF17EMP | CF16EMP | CF15EMP | CF14EMP | CF13EMP | CF12EMP | CF11EMP | CF10EMP | CF9EMP | CF8EMP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF7EMP | CF6EMP | CF5EMP | CF4EMP | CF3EMP | CF2EMP | CF1EMP | CF0EMP | RF7EMP | RF6EMP | RF5EMP | RF4EMP | RF3EMP | RF2EMP | RF1EMP | RF0EMP |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.142 RSCFDnCFDFESTS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset is returned. |
| 25 | CF17EMP | Transmit/Receive FIFO Buffer Empty Status Flag<br>0: Transmit/receive FIFO buffer k contains a message.<br>1: Transmit/receive FIFO buffer k contains no message.<br>(k = 0 to 17) |
| 24 | CF16EMP | |
| 23 | CF15EMP | |
| 22 | CF14EMP | |
| 21 | CF13EMP | |
| 20 | CF12EMP | |
| 19 | CF11EMP | |
| 18 | CF10EMP | |
| 17 | CF9EMP | |
| 16 | CF8EMP | |
| 15 | CF7EMP | |
| 14 | CF6EMP | |
| 13 | CF5EMP | |
| 12 | CF4EMP | |
| 11 | CF3EMP | |
| 10 | CF2EMP | |
| 9 | CF1EMP | |
| 8 | CF0EMP | |

Table 21.142 RSCFDnCFDFESTS Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RF7EMP | Receive FIFO Buffer Empty Status Flag<br>0: Receive FIFO buffer x contains an unread message.<br>1: Receive FIFO buffer x contains no unread message (buffer empty).<br>(x = 0 to 7) |
| 6 | RF6EMP | |
| 5 | RF5EMP | |
| 4 | RF4EMP | |
| 3 | RF3EMP | |
| 2 | RF2EMP | |
| 1 | RF1EMP | |
| 0 | RF0EMP | |

The RSCFDnCFDFESTS register is set to 03FF $FFFF_H$ in global reset mode.

### CFkEMP Flag (k = 0 to 17)

The CFkEMP flag is set to 1 when the CFEMP flag in the RSCFDnCFDCFSTSk register is set to 1 (the transmit/receive FIFO buffer contains no message (buffer empty)). When the CFEMP flag is cleared to 0 (the transmit/receive FIFO buffer contains message), the CFkEMP flag is cleared to 0.

### RFxEMP Flag (x = 0 to 7)

The RFxEMP flag is set to 1 when the RFEMP flag in the RSCFDnCFDRFSTSx register is set to 1 (the receive FIFO buffer contains no unread message). When the RFEMP flag is cleared to 0 (the receive FIFO buffer contains unread message), the RFxEMP flag is cleared to 0.

### 21.5.9.2 RSCFDnCFDFFSTS — FIFO Full Status Register

**Access:** RSCFDnCFDFFSTS register is a read-only register that can be read in 32-bit units
RSCFDnCFDFFSTSL, RSCFDnCFDFFSTSH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDFFSTSLL, RSCFDnCFDFFSTSLH, RSCFDnCFDFFSTSHL, RSCFDnCFDFFSTSHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDFFSTS: <RSCANn_base> + $023C_H$

RSCFDnCFDFFSTSL: <RSCANn_base> + $023C_H$,
RSCFDnCFDFFSTSH: <RSCANn_base> + $023E_H$

RSCFDnCFDFFSTSLL: <RSCANn_base> + $023C_H$,
RSCFDnCFDFFSTSLH: <RSCANn_base> + $023D_H$,
RSCFDnCFDFFSTSHL: <RSCANn_base> + $023E_H$,
RSCFDnCFDFFSTSHH: <RSCANn_base> + $023F_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CF17FLL | CF16FLL | CF15FLL | CF14FLL | CF13FLL | CF12FLL | CF11FLL | CF10FLL | CF9FLL | CF8FLL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF7FLL | CF6FLL | CF5FLL | CF4FLL | CF3FLL | CF2FLL | CF1FLL | CF0FLL | RF7FLL | RF6FLL | RF5FLL | RF4FLL | RF3FLL | RF2FLL | RF1FLL | RF0FLL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.143 RSCFDnCFDFFSTS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset is returned. |
| 25 | CF17FLL | Transmit/Receive FIFO Buffer Full Status Flag<br>0: Transmit/receive buffer k is not full.<br>1: Transmit/receive buffer k is full.<br>(k = 0 to 17) |
| 24 | CF16FLL | |
| 23 | CF15FLL | |
| 22 | CF14FLL | |
| 21 | CF13FLL | |
| 20 | CF12FLL | |
| 19 | CF11FLL | |
| 18 | CF10FLL | |
| 17 | CF9FLL | |
| 16 | CF8FLL | |
| 15 | CF7FLL | |
| 14 | CF6FLL | |
| 13 | CF5FLL | |
| 12 | CF4FLL | |
| 11 | CF3FLL | |
| 10 | CF2FLL | |
| 9 | CF1FLL | |
| 8 | CF0FLL | |

Table 21.143 RSCFDnCFDFFSTS Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RF7FLL | Receive FIFO Buffer Full Status Flag<br>0: Receive FIFO buffer x is not full.<br>1: Receive FIFO buffer x is full.<br>(x = 0 to 7) |
| 6 | RF6FLL | |
| 5 | RF5FLL | |
| 4 | RF4FLL | |
| 3 | RF3FLL | |
| 2 | RF2FLL | |
| 1 | RF1FLL | |
| 0 | RF0FLL | |

The RSCFDnCFDFFSTS register is cleared to 0000 $0000_H$ in global reset mode.

### CFkFLL Flag (k = 0 to 17)

The CFkFLL flag is set to 1 when the CFFLL flag in the RSCFDnCFDCFSTSk register is set to 1 (the transmit/receive FIFO buffer is full).
When the CFFLL flag is cleared to 0 (the transmit/receive FIFO buffer is not full), the CFkFLL flag is cleared to 0.

### RFxFLL Flag (x = 0 to 7)

The RFxFLL flag is set to 1 when the RFFLL flag in the RSCFDnCFDRFSTSx register is set to 1 (the receive FIFO buffer is full). When the RFFLL flag is cleared to 0 (the receive FIFO buffer is not full), the RFxFLL flag is cleared to 0.

### 21.5.9.3 RSCFDnCFDFMSTS — FIFO Message Lost Status Register

**Access:** RSCFDnCFDFMSTS register is a read-only register that can be read in 32-bit units
RSCFDnCFDFMSTSL, RSCFDnCFDFMSTSH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDFMSTSLL, RSCFDnCFDFMSTSLH, RSCFDnCFDFMSTSHL, RSCFDnCFDFMSTSHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDFMSTS: <RSCANn_base> + $0240_H$

RSCFDnCFDFMSTSL: <RSCANn_base> + $0240_H$,
RSCFDnCFDFMSTSH: <RSCANn_base> + $0242_H$

RSCFDnCFDFMSTSLL: <RSCANn_base> + $0240_H$,
RSCFDnCFDFMSTSLH: <RSCANn_base> + $0241_H$,
RSCFDnCFDFMSTSHL: <RSCANn_base> + $0242_H$,
RSCFDnCFDFMSTSHH: <RSCANn_base> + $0243_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CF17MLT | CF16MLT | CF15MLT | CF14MLT | CF13MLT | CF12MLT | CF11MLT | CF10MLT | CF9MLT | CF8MLT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF7MLT | CF6MLT | CF5MLT | CF4MLT | CF3MLT | CF2MLT | CF1MLT | CF0MLT | RF7MLT | RF6MLT | RF5MLT | RF4MLT | RF3MLT | RF2MLT | RF1MLT | RF0MLT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.144 RSCFDnCFDFMSTS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset is returned. |
| 25 | CF17MLT | Transmit/Receive FIFO Buffer Message Lost Status Flag<br>0: No transmit/receive FIFO buffer k message is lost.<br>1: A transmit/receive FIFO buffer k message is lost.<br>(k = 0 to 17) |
| 24 | CF16MLT | |
| 23 | CF15MLT | |
| 22 | CF14MLT | |
| 21 | CF13MLT | |
| 20 | CF12MLT | |
| 19 | CF11MLT | |
| 18 | CF10MLT | |
| 17 | CF9MLT | |
| 16 | CF8MLT | |
| 15 | CF7MLT | |
| 14 | CF6MLT | |
| 13 | CF5MLT | |
| 12 | CF4MLT | |
| 11 | CF3MLT | |
| 10 | CF2MLT | |
| 9 | CF1MLT | |
| 8 | CF0MLT | |

Table 21.144 RSCFDnCFDFMSTS Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RF7MLT | Receive FIFO Buffer Message Lost Status Flag<br>0: No receive FIFO buffer x message is lost.<br>1: A receive FIFO buffer x message is lost.<br>(x = 0 to 7) |
| 6 | RF6MLT | |
| 5 | RF5MLT | |
| 4 | RF4MLT | |
| 3 | RF3MLT | |
| 2 | RF2MLT | |
| 1 | RF1MLT | |
| 0 | RF0MLT | |

The RSCFDnCFDFMSTS register is cleared to 0000 $0000_H$ in global reset mode.

### CFkMLT Flag (k = 0 to 17)

The CFkMLT flag is set to 1 when the CFMLT flag in the RSCFDnCFDCFSTSk register is set to 1 (a transmit/receive FIFO message is lost). When the CFMLT flag is cleared to 0, the CFkMLT flag is cleared to 0.

### RFxMLT Flag (x = 0 to 7)

The RFxMLT flag is set to 1 when the RFMLT flag in the RSCFDnCFDRFSTSx register is set to 1 (a receive FIFO message is lost). When the RFMLT flag is cleared to 0, the RFxMLT flag is cleared to 0.

### 21.5.9.4 RSCFDnCFDRFISTS — Receive FIFO Buffer Interrupt Flag Status Register

**Access:** RSCFDnCFDRFISTS register is a read-only register that can be read in 32-bit units
RSCFDnCFDRFISTSL register is a read-only register that can be read in 16-bit units
RSCFDnCFDRFISTSLL register is a read-only register that can be read in 8-bit units

**Address:** RSCFDnCFDRFISTS: <RSCANn_base> + $0244_H$

RSCFDnCFDRFISTSL: <RSCANn_base> + $0244_H$

RSCFDnCFDRFISTSLL: <RSCANn_base> + $0244_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RF7IF | RF6IF | RF5IF | RF4IF | RF3IF | RF2IF | RF1IF | RF0IF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.145 RSCFDnCFDRFISTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned. |
| 7 | RF7IF | Receive FIFO Buffer Interrupt Request Status Flag<br>0: No receive FIFO buffer x interrupt request is present.<br>1: A receive FIFO buffer x interrupt request is present.<br>(x = 0 to 7) |
| 6 | RF6IF | |
| 5 | RF5IF | |
| 4 | RF4IF | |
| 3 | RF3IF | |
| 2 | RF2IF | |
| 1 | RF1IF | |
| 0 | RF0IF | |

The RSCFDnCFDRFISTS register is cleared to $0000\ 0000_H$ in global reset mode.

#### RFxIF Flag (x = 0 to 7)

The RFxIF flag is set to 1 when the RFIF flag in the RSCFDnCFDRFSTSx register is set to 1 (a receive FIFO interrupt request is present). When the RFIF flag is cleared to 0, the RFxIF flag is cleared to 0.

### 21.5.9.5 RSCFDnCFDCFRISTS — Transmit/receive FIFO Buffer Receive Interrupt Flag Status Register

**Access:** RSCFDnCFDCFRISTS register is a read-only register that can be read in 32-bit units
RSCFDnCFDCFRISTSL, RSCFDnCFDCFRISTSH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDCFRISTSLL, RSCFDnCFDCFRISTSLH, RSCFDnCFDCFRISTSHL registers are read-only registers that can be ready in 8-bit units

**Address:** RSCFDnCFDCFRISTS: <RSCANn_base> + $0248_H$

RSCFDnCFDCFRISTSL: <RSCANn_base> + $0248_H$,
RSCFDnCFDCFRISTSH: <RSCANn_base> + $024A_H$

RSCFDnCFDCFRISTSLL: <RSCANn_base> + $0248_H$,
RSCFDnCFDCFRISTSLH: <RSCANn_base> + $0249_H$,
RSCFDnCFDCFRISTSHL: <RSCANn_base> + $024A_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CF17RXIF | CF16RXIF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF15RXIF | CF14RXIF | CF13RXIF | CF12RXIF | CF11RXIF | CF10RXIF | CF9RXIF | CF8RXIF | CF7RXIF | CF6RXIF | CF5RXIF | CF4RXIF | CF3RXIF | CF2RXIF | CF1RXIF | CF0RXIF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.146 RSCFDnCFDCFRISTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 18 | Reserved | When read, the value after reset is returned. |
| 17 | CF17RXIF | Transmit/Receive FIFO Buffer Receive Interrupt Request Status Flag<br>0: No transmit/receive FIFO buffer k receive interrupt request is present.<br>1: A transmit/receive FIFO buffer k receive interrupt request is present.<br>(k = 0 to 17) |
| 16 | CF16RXIF | |
| 15 | CF15RXIF | |
| 14 | CF14RXIF | |
| 13 | CF13RXIF | |
| 12 | CF12RXIF | |
| 11 | CF11RXIF | |
| 10 | CF10RXIF | |
| 9 | CF9RXIF | |
| 8 | CF8RXIF | |
| 7 | CF7RXIF | |
| 6 | CF6RXIF | |
| 5 | CF5RXIF | |
| 4 | CF4RXIF | |
| 3 | CF3RXIF | |
| 2 | CF2RXIF | |
| 1 | CF1RXIF | |
| 0 | CF0RXIF | |

The RSCFDnCFDCFRISTS register is cleared to $0000\ 0000_H$ in global reset mode.

**CFkRXIF Flag (k = 0 to 17)**

The CFkRXIF flag is set to 1 when the CFRXIF flag in the RSCFDnCFDCFSTSk register is set to 1 (a transmit/receive FIFO receive interrupt request is present). When the CFRXIF flag is cleared to 0, the CFkRXIF flag is cleared to 0.

### 21.5.9.6 RSCFDnCFDCFTISTS — Transmit/receive FIFO Buffer Transmit Interrupt Flag Status Register

**Access:** RSCFDnCFDCFTISTS register is a read-only register that can be read in 32-bit units
RSCFDnCFDCFTISTSL, RSCFDnCFDCFTISTSH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDCFTISTSLL, RSCFDnCFDCFTISTSLH, RSCFDnCFDCFTISTSHL registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDCFTISTS: <RSCANn_base> + $024C_H$

RSCFDnCFDCFTISTSL: <RSCANn_base> + $024C_H$,
RSCFDnCFDCFTISTSH: <RSCANn_base> + $024E_H$

RSCFDnCFDCFTISTSLL: <RSCANn_base> + $024C_H$,
RSCFDnCFDCFTISTSLH: <RSCANn_base> + $024D_H$,
RSCFDnCFDCFTISTSHL: <RSCANn_base> + $024E_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CF17TXIF | CF16TXIF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF15TXIF | CF14TXIF | CF13TXIF | CF12TXIF | CF11TXIF | CF10TXIF | CF9TXIF | CF8TXIF | CF7TXIF | CF6TXIF | CF5TXIF | CF4TXIF | CF3TXIF | CF2TXIF | CF1TXIF | CF0TXIF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.147 RSCFDnCFDCFTISTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 18 | Reserved | When read, the value after reset is returned. |
| 17 | CF17TXIF | Transmit/Receive FIFO Buffer Transmit Interrupt Request Status Flag<br>0: No transmit/receive FIFO buffer k transmit interrupt request is present.<br>1: A transmit/receive FIFO buffer k transmit interrupt request is present.<br>(k = 0 to 17) |
| 16 | CF16TXIF | |
| 15 | CF15TXIF | |
| 14 | CF14TXIF | |
| 13 | CF13TXIF | |
| 12 | CF12TXIF | |
| 11 | CF11TXIF | |
| 10 | CF10TXIF | |
| 9 | CF9TXIF | |
| 8 | CF8TXIF | |
| 7 | CF7TXIF | |
| 6 | CF6TXIF | |
| 5 | CF5TXIF | |
| 4 | CF4TXIF | |
| 3 | CF3TXIF | |
| 2 | CF2TXIF | |
| 1 | CF1TXIF | |
| 0 | CF0TXIF | |

The RSCFDnCFDCFTISTS register is cleared to $0000\ 0000_H$ in global reset mode.

**CFkTXIF Flag (k = 0 to 17)**

The CFkTXIF flag is set to 1 when the CFTXIF flag in the RSCFDnCFDCFSTSk register is set to 1 (a transmit/receive FIFO transmit interrupt request is present). When the CFTXIF flag is cleared to 0, the CFkTXIF flag is cleared to 0.

## 21.5.10 Details of FIFO DMA-Related Registers

### 21.5.10.1 RSCFDnCFDCDTCT — DMA Enable Register

**Access:** RSCFDnCFDCDTCT register can be read or written in 32-bit units
RSCFDnCFDCDTCTL register can be read or written in 16-bit units
RSCFDnCFDCDTCTLL, RSCFDnCFDCDTCTLH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDCDTCT: <RSCANn_base> + $0490_H$

RSCFDnCFDCDTCTL: <RSCANn_base> + $0490_H$

RSCFDnCFDCDTCTLL: <RSCANn_base> + $0490_H$,
RSCFDnCFDCDTCTLH: <RSCANn_base> + $0491_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | CFDMAE5 | CFDMAE4 | CFDMAE3 | CFDMAE2 | CFDMAE1 | CFDMAE0 | RFDMAE7 | RFDMAE6 | RFDMAE5 | RFDMAE4 | RFDMAE3 | RFDMAE2 | RFDMAE1 | RFDMAE0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.148 RSCFDnCFDCDTCT Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 14 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 13 | CFDMAE5 | Transmit/Receive FIFO Buffer 15 DMA Enable<br>0: A DMA transfer request of transmit/receive FIFO buffer 15 is disabled.<br>1: A DMA transfer request of transmit/receive FIFO buffer 15 is enabled. |
| 12 | CFDMAE4 | Transmit/Receive FIFO Buffer 12 DMA Enable<br>0: A DMA transfer request of transmit/receive FIFO buffer 12 is disabled.<br>1: A DMA transfer request of transmit/receive FIFO buffer 12 is enabled. |
| 11 | CFDMAE3 | Transmit/Receive FIFO Buffer 9 DMA Enable<br>0: A DMA transfer request of transmit/receive FIFO buffer 9 is disabled.<br>1: A DMA transfer request of transmit/receive FIFO buffer 9 is enabled. |
| 10 | CFDMAE2 | Transmit/Receive FIFO Buffer 6 DMA Enable<br>0: A DMA transfer request of transmit/receive FIFO buffer 6 is disabled.<br>1: A DMA transfer request of transmit/receive FIFO buffer 6 is enabled. |
| 9 | CFDMAE1 | Transmit/Receive FIFO Buffer 3 DMA Enable<br>0: A DMA transfer request of transmit/receive FIFO buffer 3 is disabled.<br>1: A DMA transfer request of transmit/receive FIFO buffer 3 is enabled. |
| 8 | CFDMAE0 | Transmit/Receive FIFO Buffer 0 DMA Enable<br>0: A DMA transfer request of transmit/receive FIFO buffer 0 is disabled.<br>1: A DMA transfer request of transmit/receive FIFO buffer 0 is enabled. |

Table 21.148 RSCFDnCFDCDTCT Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RFDMAE7 | Receive FIFO Buffer x DMA Enable<br>0: A DMA transfer request of receive FIFO buffer x is disabled.<br>1: A DMA transfer request of receive FIFO buffer x is enabled.<br>(x = 0 to 7) |
| 6 | RFDMAE6 | |
| 5 | RFDMAE5 | |
| 4 | RFDMAE4 | |
| 3 | RFDMAE3 | |
| 2 | RFDMAE2 | |
| 1 | RFDMAE1 | |
| 0 | RFDMAE0 | |

Modify the RSCFDnCFDCDTCT register in global operating mode or global test mode.

### CFDMAEm Bit

This bit is used to enable DMA transfer for transmit/receive FIFO buffer 3 × m (the first transmit/receive FIFO buffer allocated to channel m). DMA transfer is enabled only for transmit/receive FIFO buffers for which the CFM[1:0] bits in the RSCFDnCFDCFCCk register is set to $00_B$ (receive mode). Set this bit to 0 when the CFM[1:0] value is $01_B$ (transmit mode) or $10_B$ (gateway mode).

### RFDMAEx Bit

This bit is used to enable DMA transfer for receive FIFO buffer x.

### 21.5.10.2 RSCFDnCFDCDTSTS — DMA Status Register

**Access:** RSCFDnCFDCDTSTS register is a read-only register that can be read in 32-bit units
RSCFDnCFDCDTSTSL register is a read-only register that can be read in 16-bit units
RSCFDnCFDCDTSTSLL, RSCFDnCFDCDTSTSLH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDCDTSTS: <RSCANn_base> + $0494_H$

RSCFDnCFDCDTSTSL: <RSCANn_base> + $0494_H$

RSCFDnCFDCDTSTSLL: <RSCANn_base> + $0494_H$,
RSCFDnCFDCDTSTSLH: <RSCANn_base> + $0495_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | CFDMASTS5 | CFDMASTS4 | CFDMASTS3 | CFDMASTS2 | CFDMASTS1 | CFDMASTS0 | RFDMASTS7 | RFDMASTS6 | RFDMASTS5 | RFDMASTS4 | RFDMASTS3 | RFDMASTS2 | RFDMASTS1 | RFDMASTS0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.149 RSCFDnCFDCDTSTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 14 | Reserved | When read, the value after reset is returned. |
| 13 | CFDMASTS5 | Transmit/Receive FIFO Buffer 15 DMA Status<br>0: DMA transfer of transmit/receive FIFO buffer 15 is not in progress.<br>1: DMA transfer of transmit/receive FIFO buffer 15 is in progress. |
| 12 | CFDMASTS4 | Transmit/Receive FIFO Buffer 12 DMA Status<br>0: DMA transfer of transmit/receive FIFO buffer 12 is not in progress.<br>1: DMA transfer of transmit/receive FIFO buffer 12 is in progress. |
| 11 | CFDMASTS3 | Transmit/Receive FIFO Buffer 9 DMA Status<br>0: DMA transfer of transmit/receive FIFO buffer 9 is not in progress.<br>1: DMA transfer of transmit/receive FIFO buffer 9 is in progress. |
| 10 | CFDMASTS2 | Transmit/Receive FIFO Buffer 6 DMA Status<br>0: DMA transfer of transmit/receive FIFO buffer 6 is not in progress.<br>1: DMA transfer of transmit/receive FIFO buffer 6 is in progress. |
| 9 | CFDMASTS1 | Transmit/Receive FIFO Buffer 3 DMA Status<br>0: DMA transfer of transmit/receive FIFO buffer 3 is not in progress.<br>1: DMA transfer of transmit/receive FIFO buffer 3 is in progress. |
| 8 | CFDMASTS0 | Transmit/Receive FIFO Buffer 0 DMA Status<br>0: DMA transfer of transmit/receive FIFO buffer 0 is not in progress.<br>1: DMA transfer of transmit/receive FIFO buffer 0 is in progress. |
| 7 | RFDMASTS7 | Receive FIFO Buffer x DMA Status<br>0: DMA transfer of receive FIFO buffer x is not in progress.<br>1: DMA transfer of receive FIFO buffer x is in progress.<br>(x = 0 to 7) |
| 6 | RFDMASTS6 | |
| 5 | RFDMASTS5 | |
| 4 | RFDMASTS4 | |
| 3 | RFDMASTS3 | |
| 2 | RFDMASTS2 | |
| 1 | RFDMASTS1 | |
| 0 | RFDMASTS0 | |

#### CFDMASTSm Bit

When DMA transfer is enabled (CFDMAEm bit in the RSCFDnCFDCDTCT register is 1) for the transmit/receive FIFO buffer 3 × m (the first transmit/receive FIFO buffer allocated to channel m) while the transmit/receive FIFO buffer contains one of more messages, the CFDMASTSm bit is set to 1 indicating that DMA transfer is in progress.

When all messages in the transmit/receive FIFO buffer have been transferred or DMA transfer is disabled (CFDMAEm bit is 0), the CFDMASTSm bit is cleared to 0 indicating that DMA transfer has been completed. If the CFDMAEm bit is set to 0 during DMA transfer, the CFDMASTSm bit is cleared to 0 after the ongoing DMA transfer has been completed (when the message that is being transferred has been transferred to the last byte in the payload storage area).

These bits are cleared to 0 in global reset mode.

#### RFDMASTSx Bit

When DMA transfer is enabled (corresponding RFDMAEx bit in the RSCFDnCFDCDTCT register is 1) for the receive FIFO buffer x and the receive FIFO buffer contains one of more messages, the RFDMAEx bit is set to 1 indicating that DMA transfer is in progress.

When all messages in the receive FIFO buffer x have been transferred or DMA transfer is disabled (RFDMAEx bit = 0), the RFDMASTSx bit is cleared to 0 indicating that DMA transfer has been completed. If the RFDMAEx bit is set to 0 during DMA transfer, the RFDMASTSx bit is cleared to 0 after the ongoing DMA transfer has been completed (when the message that is being transferred has been transferred to the last byte in the payload storage area).

These bits are cleared to 0 in global reset mode.

## 21.5.11 Details of Transmit Buffer-related Registers

### 21.5.11.1 RSCFDnCFDTMCp — Transmit Buffer Control Register (p = 0 to 95)

**Access:** RSCFDnCFDTMCp register can be read or written in 8-bit units

**Address:** RSCFDnCFDTMCp: <RSCANn_base> + $0250_H$ + ($01_H$ × p)

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TMOM | TMTAR | TMTR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W*1 | R/W*1 |

Note 1. The only effective value for writing to this bit is 1, which sets the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.150 RSCFDnCFDTMCp Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 | TMOM | One-Shot Transmission Enable<br>0: One-shot transmission is disabled.<br>1: One-shot transmission is enabled. |
| 1 | TMTAR | Transmit Abort Request<br>0: Transmit abort is not requested.<br>1: Transmit abort is requested. |
| 0 | TMTR | Transmit Request<br>0: Transmission is not requested.<br>1: Transmission is requested. |

When the RSCFDnCFDTMCp register meets any of the following conditions, set it to $00_H$.

- The RSCFDnCFDTMCp register corresponds to the transmit buffer number selected by the CFTML[3:0] bits in the RSCFDnCFDCFCCk register (p = m × 16 + the value of CFTML[3:0] bits).
- The RSCFDnCFDTMCp register corresponds to the transmit buffer allocated to the transmit queue by the TXQDC[3:0] bits in the RSCFDnCFDTXQCCm (m = 0 to 5) register (p = (m × 16 + 15) to (m × 16 + 15 – the value of TXQDC[3:0] bits)).
- RSCFDnCFDTMCp register (p = (m × 16) + 1, (m × 16) + 2, (m × 16) + 4, or (m × 16) + 5) corresponding to the transmit buffer allocated as a payload storage area when the TMME bit in the RSCFDnCFDCmFDCFG register is 1 (transmit buffer merge mode)

Bits in the RSCFDnCFDTMCp register are all cleared to 0 in channel reset mode. Modify the RSCFDnCFDTMCp register in channel communication mode or channel halt mode.

#### TMOM Bit

Setting this bit to 1 enables one-shot transmission. When transmission fails, retransmission defined in the CAN protocol is not performed.

Modify the TMOM bit when the TMTRM flag in the RSCFDnCFDTMSTSp register is set to 0. Set the TMOM bit to 1 together with the TMTR bit.

#### TMTAR Bit

Setting this bit to 1 generates a transmit abort request for the message stored in the transmit buffer. However, a message that is being transmitted or the one that will be transmitted next cannot be aborted.

The TMTAR bit can be set to 1 when TMTR bit is 1.

The TMTAR bit is cleared to 0 when any of the following conditions is met, but cannot be cleared by the program writing 0 to the bit.

- Transmission has been completed.
- Transmit abort has been completed.
- An error or arbitration lost has been detected.

If this bit becomes 0 at the same time as the program writes 1 to this bit, this bit becomes 0.

#### TMTR Bit

Setting this bit to 1 transmits the message stored in the transmit buffer.

The TMTR bit is cleared to 0 when any of the following conditions is met, but cannot be cleared by the program writing 0 to the bit.

- Transmission has been completed.
- Transmit abort has been completed after the TMTAR bit was set to 1.
- An error or arbitration-lost has been detected with the TMOM bit set to 1.

Set the TMTR bit to 1 when the value of TMTRF[1:0] in the RSCFDnCFDTMSTSp register is $00_B$.

### 21.5.11.2 RSCFDnCFDTMSTSp — Transmit Buffer Status Register (p = 0 to 95)

**Access:** RSCFDnCFDTMSTSp register can be read or written in 8-bit units

**Address:** RSCFDnCFDTMSTSp: <RSCANn_base> + $02D0_H$ + ($01_H$ × p)

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | TMTARM | TMTRM | TMTRF[1:0] | | TMTSTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R |

**Table 21.151 RSCFDnCFDTMSTSp Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 5 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 4 | TMTARM | Transmit Buffer Transmit Abort Request Status Flag<br>0: No transmit abort request is present.<br>1: A transmit abort request is present. |
| 3 | TMTRM | Transmit Buffer Transmit Request Status Flag<br>0: No transmit request is present.<br>1: A transmit request is present. |
| 2, 1 | TMTRF[1:0] | Transmit Buffer Transmit Result Status Flag<br>b2 b1<br>0 0: Transmission is in progress or no transmit request is present.<br>0 1: Transmit abort has been completed.<br>1 0: Transmission has been completed (without transmit abort request).<br>1 1: Transmission has been completed (with transmit abort request). |
| 0 | TMTSTS | Transmit Buffer Transmit Status Flag<br>0: Transmission is not in progress.<br>1: Transmission is in progress. |

The RSCFDnCFDTMSTSp register is cleared to all 0 in channel reset mode.

#### TMTARM Flag

The TMTARM flag is set to 1 when the TMTAR bit in the RSCFDnCFDTMCp register is set to 1.

The TMTARM flag is set to 0 when the TMTAR bit in the RSCFDnCFDTMCp register is set to 0.

#### TMTRM Flag

The TMTRM flag is set to 1 when the TMTR bit in the RSCFDnCFDTMCp register is set to 1.

The TMTRM flag is set to 0 when the TMTR bit in the RSCFDnCFDTMCp register is set to 0.

#### TMTRF[1:0] Flag

This flag indicates the result of transmission from the transmit buffer.

$00_B$: Transmission is in progress or no transmit request is present.

$01_B$: Transmission from the transmit buffer was aborted.

$10_B$: Transmission has been completed with the TMTAR bit in the RSCFDnCFDTMCp register set to 0 (transmit abort is not requested).

$11_B$: Transmission has been completed with the TMTAR bit in the RSCFDnCFDTMCp register set to 1 (transmit abort is requested).

Write $00_B$ to the TMTRF[1:0] flag in channel communication mode or channel halt mode. Do not write any value other than $00_B$ to this flag.

#### TMTSTS Flag

This flag is set to 1 when transmission from the transmit buffer starts, and is cleared to 0 when transmission from the transmit buffer has been completed or terminated due to a bus error or arbitration lost.

### 21.5.11.3 RSCFDnCFDTMIDp — Transmit Buffer ID Register (p = 0 to 95)

**Access:** RSCFDnCFDTMIDp register can be read or written in 32-bit units
RSCFDnCFDTMIDpL, RSCFDnCFDTMIDpH registers can be read or written in 16-bit units
RSCFDnCFDTMIDpLL, RSCFDnCFDTMIDpLH, RSCFDnCFDTMIDpHL, RSCFDnCFDTMIDpHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDTMIDp: <RSCANn_base> + $4000_H$ + ($20_H$ × p)

RSCFDnCFDTMIDpL: <RSCANn_base> + $4000_H$ + ($20_H$ × p),
RSCFDnCFDTMIDpH: <RSCANn_base> + $4002_H$ + ($20_H$ × p)

RSCFDnCFDTMIDpLL: <RSCANn_base> + $4000_H$ + ($20_H$ × p),
RSCFDnCFDTMIDpLH: <RSCANn_base> + $4001_H$ + ($20_H$ × p),
RSCFDnCFDTMIDpHL: <RSCANn_base> + $4002_H$ + ($20_H$ × p),
RSCFDnCFDTMIDpHH: <RSCANn_base> + $4003_H$ + ($20_H$ × p)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMIDE | TMRTR | THLEN | TMID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.152 RSCFDnCFDTMIDp Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | TMIDE | Transmit Buffer IDE<br>0: Standard ID<br>1: Extended ID |
| 30 | TMRTR | Transmit Buffer RTR/RRS<br>• When the transmit message is a classical CAN frame<br>0: Data frame<br>1: Remote frame<br>• When the transmit message is a CAN FD frame<br>Write 0 to this bit. |
| 29 | THLEN | Transmit History Data Store Enable<br>0: Transmit history data is not stored in the buffer.<br>1: Transmit history data is stored in the buffer. |
| 28 to 0 | TMID[28:0] | Transmit Buffer ID Data<br>Set standard ID or extended ID.<br>For standard ID, write an ID to bits 10 to 0 and write 0 to bits 28 to 11. |

Modify this register when the TMTRM bit in the corresponding RSCFDnCFDTMSTSp register is set to 0 (no transmit request is present). If this register is linked to a transmit/receive FIFO buffer, do not write data to this register. If this register is allocated to the transmit queue, only write data to a transmit buffer p (p = m × 16 + 15) for the corresponding channel.

#### TMIDE Bit

This bit is used to set the ID format of the message to be transmitted from the transmit buffer.

#### TMRTR Bit

This bit is used to set the data format of the message to be transmitted from the transmit buffer.

Set this bit to 0 when the TMFDF bit in the RSCFDnCFDTMFDCTRp register is 1 (CAN FD frame).

#### THLEN Bit

When this bit is set to 1, the transmit history data (label information, buffer number, buffer type, and timestamp) of transmit messages is stored in the transmit history buffer after transmission is completed.

#### TMID[28:0] Bits

These bits are used to set the ID of the message to be transmitted from the transmit buffer.

### 21.5.11.4 RSCFDnCFDTMPTRp — Transmit Buffer Pointer Register (p = 0 to 95)

**Access:** RSCFDnCFDTMPTRp register can be read or written in 32-bit units
RSCFDnCFDTMPTRpH register can be read or written in 16-bit units
RSCFDnCFDTMPTRpHL, RSCFDnCFDTMPTRpHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDTMPTRp: <RSCANn_base> + $4004_H$ + ($20_H$ × p)

RSCFDnCFDTMPTRpH: <RSCANn_base> + $4006_H$ + ($20_H$ × p)

RSCFDnCFDTMPTRpHL: <RSCANn_base> + $4006_H$ + ($20_H$ × p),
RSCFDnCFDTMPTRpHH: <RSCANn_base> + $4007_H$ + ($20_H$ × p)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMDLC[3:0] | | | | — | — | — | — | TMPTR[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.153 RSCFDnCFDTMPTRp Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | TMDLC[3:0] | Transmit Buffer DLC Data |
| 27 to 24 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 23 to 16 | TMPTR[7:0] | Transmit Buffer Label Data<br>Set the label information to be stored in the transmit history buffer. |
| 15 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

| b31 | b30 | b29 | b28 | Classical CAN Frame | CAN FD Frame |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 data bytes | |
| 0 | 0 | 0 | 1 | 1 data byte | |
| 0 | 0 | 1 | 0 | 2 data bytes | |
| 0 | 0 | 1 | 1 | 3 data bytes | |
| 0 | 1 | 0 | 0 | 4 data bytes | |
| 0 | 1 | 0 | 1 | 5 data bytes | |
| 0 | 1 | 1 | 0 | 6 data bytes | |
| 0 | 1 | 1 | 1 | 7 data bytes | |
| 1 | 0 | 0 | 0 | 8 data bytes | |
| 1 | 0 | 0 | 1 | 8 data bytes | 12 data bytes |
| 1 | 0 | 1 | 0 | | 16 data bytes |
| 1 | 0 | 1 | 1 | | 20 data bytes |
| 1 | 1 | 0 | 0 | | 24 data bytes |
| 1 | 1 | 0 | 1 | | 32 data bytes |
| 1 | 1 | 1 | 0 | | 48 data bytes |
| 1 | 1 | 1 | 1 | | 64 data bytes |

Modify this register when the TMTRM bit in the corresponding RSCFDnCFDTMSTSp register is set to 0 (no transmit request is present). If this register is linked to a transmit/receive FIFO buffer, do not write to this register. If this register is allocated to the transmit queue, only write to a transmit buffer p (p = m × 16 + 15) for the corresponding channel.

#### TMDLC[3:0] Bits

These bits are used to set the data length of the message to be transmitted from the transmit buffer when the TMRTR bit in the RSCFDnCFDTMIDp register is set to 0 (data frame).

When the TMDLC[3:0] bits are set to $1001_B$ or more while the TMFDF bit in the RSCFDnCFDTMFDCTRp register is 0 (classical CAN frame), 8-byte data is transmitted actually. When the TMFDF bit is 1 (CAN FD frame), the settable value range varies depending on the setting of the TMME bit in the RSCFDnCFDCmFDCFG register.

- When the TMME bit = 0 (transmit buffer merge mode disabled):
  A value of $0000_B$ to $1111_B$ is settable. If a value larger than $1100_B$ is set, payloads exceeding 20 bytes are padded by CCH.
- When the TMME bit = 1 (transmit buffer merge mode enabled):
  When the corresponding transmit buffer number p = (m × 16) + 0 or (m × 16) + 3, a value of $0000_B$ to $1111_B$ is settable. In other cases, set a value of $0000_B$ to $1011_B$ (20 data bytes).

When the TMRTR bit is 1 (remote frame), set the data length of a message to be requested.

#### TMPTR[7:0] Bits

When message transmission has been completed, the TMPTR[7:0] value is stored in the transmit history buffer.

### 21.5.11.5 RSCFDnCFDTMFDCTRp — Transmit Buffer CAN FD Configuration Register (p = 0 to 95)

**Access:** RSCFDnCFDTMFDCTRp register can be read or written in 32-bit units
RSCFDnCFDTMFDCTRpL register can be read or written in 16-bit units
RSCFDnCFDTMFDCTRpLL register can be read or written in 8-bit units

**Address:** RSCFDnCFDTMFDCTRp: <RSCANn_base> + $4008_H$ + ($20_H$ × p)

RSCFDnCFDTMFDCTRpL: <RSCANn_base> + $4008_H$ + ($20_H$ × p)

RSCFDnCFDTMFDCTRpLL: <RSCANn_base> + $4008_H$ + ($20_H$ × p)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TMFDF | TMBRS | TMESI |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 21.154 RSCFDnCFDTMFDCTRp Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 | TMFDF | FDF<br>0: Classical CAN frame<br>1: CAN FD frame |
| 1 | TMBRS | BRS<br>0: The bit rate in the data area does not change.<br>1: The bit rate in the data area changes. |
| 0 | TMESI | ESI<br>0: Error active node<br>1: Error passive node |

Modify this register when the TMTRM bit in the corresponding RSCFDnCFDTMSTSp register is 0 (transmission not requested). When this register is linked to the transmit/receive FIFO buffer, do not write data to this register. When this register is allocated to the transmit queue, write data only to transmit buffer p (p = m × 16 + 15) of the corresponding channel.

#### TMFDF Bit

This bit is used to set the FD format of the message to be transmitted from the transmit buffer.

#### TMBRS Bit

When this bit is set to 1 while the TMFDF bit is 1, the data area of a transmit message is transmitted at the data bit rate. When the TMFDF bit is 0, write 0 to this bit.

**TMESI Bit**

This bit is used to set the ESI bit value of the message to be transmitted from the transmit buffer when the TMFDF bit is 1. The set value is transmitted when the ESIC bit in the RSCFDnCFDCmFDCFG register is 1 and the channel is in the error active state. When the channel is in the error passive state, the ESI bit value that shows an error passive node is transmitted regardless of this bit value. When the TMFDF bit is 0, write 0 to this bit.

### 21.5.11.6 RSCFDnCFDTMDFb_p — Transmit Buffer Data Field b Register (b = 0 to 4, p = 0 to 95)

**Access:** RSCFDnCFDTMDFb_p register can be read or written in 32-bit units
RSCFDnCFDTMDFb_pL, RSCFDnCFDTMDFb_pH registers can be read or written in 16-bit units
RSCFDnCFDTMDFb_pLL, RSCFDnCFDTMDFb_pLH, RSCFDnCFDTMDFb_pHL, RSCFDnCFDTMDFb_pHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDTMDFb_p: <RSCANn_base> + $400C_H$ + ($04_H$ × b) + ($20_H$ × p)

RSCFDnCFDTMDFb_pL: <RSCANn_base> + $400C_H$ + ($04_H$ × b) + ($20_H$ × p),
RSCFDnCFDTMDFb_pH: <RSCANn_base> + $400E_H$ + ($04_H$ × b) + ($20_H$ × p)

RSCFDnCFDTMDFb_pLL: <RSCANn_base> + $400C_H$ + ($04_H$ × b) + ($20_H$ × p),
RSCFDnCFDTMDFb_pLH: <RSCANn_base> + $400D_H$ + ($04_H$ × b) + ($20_H$ × p),
RSCFDnCFDTMDFb_pHL: <RSCANn_base> + $400E_H$ + ($04_H$ × b) + ($20_H$ × p),
RSCFDnCFDTMDFb_pHH: <RSCANn_base> + $400F_H$ + ($04_H$ × b) + ($20_H$ × p)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMDB4 × b + 3 [7:0] | | | | | | | | TMDB4 × b + 2 [7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMDB4 × b + 1 [7:0] | | | | | | | | TMDB4 × b + 0 [7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.155 RSCFDnCFDTMDFb_p Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | TMDB4 × b + 3 [7:0] | Transmit Buffer Data Byte 4 × b + 3<br>Transmit Buffer Data Byte 4 × b + 2<br>Transmit Buffer Data Byte 4 × b + 1<br>Transmit Buffer Data Byte 4 × b + 0<br><br>Set the transmit buffer data. |
| 23 to 16 | TMDB4 × b + 2 [7:0] | |
| 15 to 8 | TMDB4 × b + 1 [7:0] | |
| 7 to 0 | TMDB4 × b + 0 [7:0] | |

Modify this register when the TMTRM bit in the corresponding RSCFDnCFDTMSTSp register is set to 0 (no transmit request is present). If this register is linked to a transmit/receive FIFO buffer, do not write to this register. If this register is allocated to the transmit queue, only write to a transmit buffer p (p = m × 16 + 15) for the corresponding channel.

### 21.5.11.7 RSCFDnCFDTMIECy — Transmit Buffer Interrupt Enable Configuration Register (y = 0 to 2)

**Access:** RSCFDnCFDTMIECy register can be read or written in 32-bit units
RSCFDnCFDTMIECyL, RSCFDnCFDTMIECyH registers can be read or written in 16-bit units
RSCFDnCFDTMIECyLL, RSCFDnCFDTMIECyLH, RSCFDnCFDTMIECyHL, RSCFDnCFDTMIECyHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDTMIECy: <RSCANn_base> + $0390_H$ + ($04_H$ × y)

RSCFDnCFDTMIECyL: <RSCANn_base> + $0390_H$ + ($04_H$ × y),
RSCFDnCFDTMIECyH: <RSCANn_base> + $0392_H$ + ($04_H$ × y)

RSCFDnCFDTMIECyLL: <RSCANn_base> + $0390_H$ + ($04_H$ × y),
RSCFDnCFDTMIECyLH: <RSCANn_base> + $0391_H$ + ($04_H$ × y),
RSCFDnCFDTMIECyHL: <RSCANn_base> + $0392_H$ + ($04_H$ × y),
RSCFDnCFDTMIECyHH: <RSCANn_base> + $0393_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMIEp (p = y × 32 + 31 to y × 32 + 16 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMIEp (p = y × 32 + 15 to y × 32 + 0 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.156 RSCFDnCFDTMIECy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | TMIEp | Transmit Buffer Interrupt Enable p (p = y × 32 + 31 to y × 32 + 16)<br>0: Transmit buffer interrupt is disabled<br>1: Transmit buffer interrupt is enabled |
| 15 to 0 | TMIEp | Transmit Buffer Interrupt Enable p (p = y × 32 + 15 to y × 32 + 0)<br>0: Transmit buffer interrupt is disabled.<br>1: Transmit buffer interrupt is enabled. |

#### TMIEp Bits (p = 0 to 95)

When any of these bits is set to 1 and the corresponding transmission has been completed, a transmit buffer interrupt request is generated.

Modify these bits when the TMTRM flag in the corresponding RSCFDnCFDTMSTSp register is 0 (no transmit request is present).

Write 0 to bits corresponding to transmit buffers linked to transmit/receive FIFO buffers or transmit buffers allocated to the transmit queue.

When the TMME bit in the RSCFDnCFDCmFDCFG register is 1 (transmit buffer merge mode enable), set the bit corresponding to the transmit buffer allocated as a payload storage area to 0.

**Table 21.157** shows the bit assignment.

**Table 21.157 TMIEp Bit Assignment**

| Bit Position | Channel | Transmit Buffer Number |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| . . | . . | . . |
| 15 | 0 | 15 |
| 16 | 1 | 0 |
| . . | . . | . . |
| 30 | 1 | 14 |
| 31 | 1 | 15 |
| 32 | 2 | 0 |
| 33 | 2 | 1 |
| . . | . . | . . |
| 47 | 2 | 15 |
| 48 | 3 | 0 |
| . . | . . | . . |
| 62 | 3 | 14 |
| 63 | 3 | 15 |
| 64 | 4 | 0 |
| 65 | 4 | 1 |
| . . | . . | . . |
| 78 | 4 | 14 |
| 79 | 4 | 15 |
| 80 | 5 | 0 |
| 81 | 5 | 1 |
| . . | . . | . . |
| 94 | 5 | 14 |
| 95 | 5 | 15 |

## 21.5.12 Details of Transmit Buffer Status-related Registers

### 21.5.12.1 RSCFDnCFDTMTRSTSy — Transmit Buffer Transmit Request Status Register (y = 0 to 2)

**Access:** RSCFDnCFDTMTRSTSy register is a read-only register that can be read in 32-bit units
RSCFDnCFDTMTRSTSyL, RSCFDnCFDTMTRSTSyH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDTMTRSTSyLL, RSCFDnCFDTMTRSTSyLH, RSCFDnCFDTMTRSTSyHL, RSCFDnCFDTMTRSTSyHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDTMTRSTSy: <RSCANn_base> + $0350_H$ + ($04_H$ × y)

RSCFDnCFDTMTRSTSyL: <RSCANn_base> + $0350_H$ + ($04_H$ × y),
RSCFDnCFDTMTRSTSyH: <RSCANn_base> + $0352_H$ + ($04_H$ × y)

RSCFDnCFDTMTRSTSyLL: <RSCANn_base> + $0350_H$ + ($04_H$ × y),
RSCFDnCFDTMTRSTSyLH: <RSCANn_base> + $0351_H$ + ($04_H$ × y),
RSCFDnCFDTMTRSTSyHL: <RSCANn_base> + $0352_H$ + ($04_H$ × y),
RSCFDnCFDTMTRSTSyHH: <RSCANn_base> + $0353_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTRSTSp (p = y × 32 + 31 to y × 32 + 16 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTRSTSp (p = y × 32 + 15 to y × 32 + 0 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.158 RSCFDnCFDTMTRSTSy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | TMTRSTSp | Transmit Buffer Transmit Request Status Flag p (p = y × 32 + 31 to y × 32 + 16)<br>0: No transmit request is present.<br>1: A transmit request is present. |
| 15 to 0 | TMTRSTSp | Transmit Buffer Transmit Request Status Flag p (p = y × 32 + 15 to y × 32 + 0)<br>0: No transmit request is present.<br>1: A transmit request is present. |

#### TMTRSTSp Flags (p = 0 to 95)

These flags indicate the status of the TMTR bit in the RSCFDnCFDTMCp register.

When the TMTR bit is set to 1 (transmission is requested), the corresponding TMTRSTSp flag is set to 1.

The corresponding TMTRSTSp flag is cleared to 0 when the TMTR bit is set to 0 (transmission is not requested) or in channel reset mode.

**Table 21.159** shows the bit assignment.

**Table 21.159 TMTRSTSp Bit Assignment**

| Bit Position | Channel | Transmit Buffer Number |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| . . | . . | . . |
| 15 | 0 | 15 |
| 16 | 1 | 0 |
| . . | . . | . . |
| 30 | 1 | 14 |
| 31 | 1 | 15 |
| 32 | 2 | 0 |
| 33 | 2 | 1 |
| . . | . . | . . |
| 47 | 2 | 15 |
| 48 | 3 | 0 |
| . . | . . | . . |
| 62 | 3 | 14 |
| 63 | 3 | 15 |
| 64 | 4 | 0 |
| 65 | 4 | 1 |
| . . | . . | . . |
| 78 | 4 | 14 |
| 79 | 4 | 15 |
| 80 | 5 | 0 |
| 81 | 5 | 1 |
| . . | . . | . . |
| 94 | 5 | 14 |
| 95 | 5 | 15 |

### 21.5.12.2 RSCFDnCFDTMTARSTSy — Transmit Buffer Transmit Abort Request Status Register (y = 0 to 2)

**Access:** RSCFDnCFDTMTARSTSy register is a read-only register that can be read in 32-bit units
RSCFDnCFDTMTARSTSyL, RSCFDnCFDTMTARSTSyH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDTMTARSTSyLL, RSCFDnCFDTMTARSTSyLH, RSCFDnCFDTMTARSTSyHL, RSCFDnCFDTMTARSTSyHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDTMTARSTSy: <RSCANn_base> + $0360_H$ + ($04_H$ × y)

RSCFDnCFDTMTARSTSyL: <RSCANn_base> + $0360_H$ + ($04_H$ × y),
RSCFDnCFDTMTARSTSyH: <RSCANn_base> + $0362_H$ + ($04_H$ × y)

RSCFDnCFDTMTARSTSyLL: <RSCANn_base> + $0360_H$ + ($04_H$ × y),
RSCFDnCFDTMTARSTSyLH: <RSCANn_base> + $0361_H$ + ($04_H$ × y),
RSCFDnCFDTMTARSTSyHL: <RSCANn_base> + $0362_H$ + ($04_H$ × y),
RSCFDnCFDTMTARSTSyHH: <RSCANn_base> + $0363_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTARSTSp (p = y × 32 + 31 to y × 32 + 16 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTARSTSp (p = y × 32 + 15 to y × 32 + 0 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.160 RSCFDnCFDTMTARSTSy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | TMTARSTSp | Transmit Buffer Transmit Abort Request Status Flag p (p = y × 32 + 31 to y × 32 + 16)<br>0: No transmit abort request is present.<br>1: A transmit abort request is present. |
| 15 to 0 | TMTARSTSp | Transmit Buffer Transmit Abort Request Status Flag p (p = y × 32 + 15 to y × 32 + 0)<br>0: No transmit abort request is present.<br>1: A transmit abort request is present. |

#### TMTARSTSp Flags (p = 0 to 95)

These flags indicate the status of the TMTAR bit in the RSCFDnCFDTMCp register.

When the TMTAR bit is set to 1 (transmit abort is requested), the corresponding TMTARSTSp flag is set to 1.

The corresponding TMTARSTSp flag is cleared to 0 when the TMTAR bit is set to 0 (transmit abort is not requested) or in channel reset mode.

**Table 21.161** shows the bit assignment.

Table 21.161 TMTARSTSp Bit Assignment

| Bit Position | Channel | Transmit Buffer Number |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| . . | . . | . . |
| 15 | 0 | 15 |
| 16 | 1 | 0 |
| . . | . . | . . |
| 30 | 1 | 14 |
| 31 | 1 | 15 |
| 32 | 2 | 0 |
| 33 | 2 | 1 |
| . . | . . | . . |
| 47 | 2 | 15 |
| 48 | 3 | 0 |
| . . | . . | . . |
| 62 | 3 | 14 |
| 63 | 3 | 15 |
| 64 | 4 | 0 |
| 65 | 4 | 1 |
| . . | . . | . . |
| 78 | 4 | 14 |
| 79 | 4 | 15 |
| 80 | 5 | 0 |
| 81 | 5 | 1 |
| . . | . . | . . |
| 94 | 5 | 14 |
| 95 | 5 | 15 |

### 21.5.12.3 RSCFDnCFDTMTCSTSy — Transmit Buffer Transmit Complete Status Register (y = 0 to 2)

**Access:** RSCFDnCFDTMTCSTSy register is a read-only register that can be read in 32-bit units
RSCFDnCFDTMTCSTSyL, RSCFDnCFDTMTCSTSyH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDTMTCSTSyLL, RSCFDnCFDTMTCSTSyLH, RSCFDnCFDTMTCSTSyHL, RSCFDnCFDTMTCSTSyHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDTMTCSTSy: <RSCANn_base> + $0370_H$ + ($04_H$ × y)

RSCFDnCFDTMTCSTSyL: <RSCANn_base> + $0370_H$ + ($04_H$ × y),
RSCFDnCFDTMTCSTSyH: <RSCANn_base> + $0372_H$ + ($04_H$ × y)

RSCFDnCFDTMTCSTSyLL: <RSCANn_base> + $0370_H$ + ($04_H$ × y),
RSCFDnCFDTMTCSTSyLH: <RSCANn_base> + $0371_H$ + ($04_H$ × y),
RSCFDnCFDTMTCSTSyHL: <RSCANn_base> + $0372_H$ + ($04_H$ × y),
RSCFDnCFDTMTCSTSyHH: <RSCANn_base> + $0373_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTCSTSp (p = y × 32 + 31 to y × 32 + 16 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTCSTSp (p = y × 32 + 15 to y × 32 + 0 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.162 RSCFDnCFDTMTCSTSy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | TMTCSTSp | Transmit Buffer Transmit Complete Status Flag p (p = y × 32 + 31 to y × 32 + 16)<br>0: Transmission has not been completed.<br>1: Transmission has been completed. |
| 15 to 0 | TMTCSTSp | Transmit Buffer Transmit Complete Status Flag p (p = y × 32 + 15 to y × 32 + 0)<br>0: Transmission has not been completed.<br>1: Transmission has been completed. |

#### TMTCSTSp Flags (p = 0 to 95)

When the TMTRF[1:0] flag in the RSCFDnCFDTMSTSp register is set to $10_B$ (transmission has been completed (without transmit abort request)) or $11_B$ (transmission has been completed (with transmit abort request)), the corresponding TMTCSTSp flag is set to 1.

A TMTCSTSp flag is cleared to 0 when the corresponding TMTRF[1:0] flag is set to $00_B$ or in channel reset mode.

**Table 21.163** shows the bit assignment.

**Table 21.163 TMTCSTSp Bit Assignment**

| Bit Position | Channel | Transmit Buffer Number |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| . . | . . | . . |
| 15 | 0 | 15 |
| 16 | 1 | 0 |
| . . | . . | . . |
| 30 | 1 | 14 |
| 31 | 1 | 15 |
| 32 | 2 | 0 |
| 33 | 2 | 1 |
| . . | . . | . . |
| 47 | 2 | 15 |
| 48 | 3 | 0 |
| . . | . . | . . |
| 62 | 3 | 14 |
| 63 | 3 | 15 |
| 64 | 4 | 0 |
| 65 | 4 | 1 |
| . . | . . | . . |
| 78 | 4 | 14 |
| 79 | 4 | 15 |
| 80 | 5 | 0 |
| 81 | 5 | 1 |
| . . | . . | . . |
| 94 | 5 | 14 |
| 95 | 5 | 15 |

### 21.5.12.4 RSCFDnCFDTMTASTSy — Transmit Buffer Transmit Abort Status Register (y = 0 to 2)

**Access:** RSCFDnCFDTMTASTSy register is a read-only register that can be read in 32-bit units
RSCFDnCFDTMTASTSyL, RSCFDnCFDTMTASTSyH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDTMTASTSyLL, RSCFDnCFDTMTASTSyLH, RSCFDnCFDTMTASTSyHL, RSCFDnCFDTMTASTSyHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDTMTASTSy: <RSCANn_base> + $0380_H$ + ($04_H$ × y)

RSCFDnCFDTMTASTSyL: <RSCANn_base> + $0380_H$ + ($04_H$ × y),
RSCFDnCFDTMTASTSyH: <RSCANn_base> + $0382_H$ + ($04_H$ × y)

RSCFDnCFDTMTASTSyLL: <RSCANn_base> + $0380_H$ + ($04_H$ × y),
RSCFDnCFDTMTASTSyLH: <RSCANn_base> + $0381_H$ + ($04_H$ × y),
RSCFDnCFDTMTASTSyHL: <RSCANn_base> + $0382_H$ + ($04_H$ × y),
RSCFDnCFDTMTASTSyHH: <RSCANn_base> + $0383_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTASTSp (p = y × 32 + 31 to y × 32 + 16 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTASTSp (p = y × 32 + 15 to y × 32 + 0 (y = 0, 1, 2)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.164 RSCFDnCFDTMTASTSy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | TMTASTSp | Transmit Buffer Transmit Abort Status Flag p (p = y × 32 + 31 to y × 32 + 16)<br>0: Transmission is not aborted<br>1: Transmission is aborted |
| 15 to 0 | TMTASTSp | Transmit Buffer Transmit Abort Status Flag p (p = y × 32 + 15 to y × 32 + 0)<br>0: Transmission is not aborted.<br>1: Transmission is aborted. |

#### TMTASTSp Flags (p = 0 to 95)

When the TMTRF[1:0] flag in the RSCFDnCFDTMSTSp register is set to $01_B$ (transmit abort has been completed), the corresponding TMTASTSp flag is set to 1.

A TMTASTSp flag is cleared to 0 when the corresponding TMTRF[1:0] flag is set to $00_B$ or in channel reset mode.

**Table 21.165** shows the bit assignment.

**Table 21.165 TMTASTSp Bit Assignment**

| Bit Position | Channel | Transmit Buffer Number |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| . . | . . | . . |
| 15 | 0 | 15 |
| 16 | 1 | 0 |
| . . | . . | . . |
| 30 | 1 | 14 |
| 31 | 1 | 15 |
| 32 | 2 | 0 |
| 33 | 2 | 1 |
| . . | . . | . . |
| 47 | 2 | 15 |
| 48 | 3 | 0 |
| . . | . . | . . |
| 62 | 3 | 14 |
| 63 | 3 | 15 |
| 64 | 4 | 0 |
| 65 | 4 | 1 |
| . . | . . | . . |
| 78 | 4 | 14 |
| 79 | 4 | 15 |
| 80 | 5 | 0 |
| 81 | 5 | 1 |
| . . | . . | . . |
| 94 | 5 | 14 |
| 95 | 5 | 15 |

## 21.5.13 Details of Transmit Queue-related Registers

### 21.5.13.1 RSCFDnCFDTXQCCm — Transmit Queue Configuration and Control Register (m = 0 to 5)

**Access:** RSCFDnCFDTXQCCm register can be read or written in 32-bit units
RSCFDnCFDTXQCCmL register can be read or written in 16-bit units
RSCFDnCFDTXQCCmLL, RSCFDnCFDTXQCCmLH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDTXQCCm: <RSCANn_base> + $03A0_H$ + ($04_H$ × m)

RSCFDnCFDTXQCCmL: <RSCANn_base> + $03A0_H$ + ($04_H$ × m)

RSCFDnCFDTXQCCmLL: <RSCANn_base> + $03A0_H$ + ($04_H$ × m),
RSCFDnCFDTXQCCmLH: <RSCANn_base> + $03A1_H$ + ($04_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | TXQIM | TXQIE | TXQDC[3:0] | | | | — | — | — | — | — | — | — | TXQE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W |

**Table 21.166 RSCFDnCFDTXQCCm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 14 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 13 | TXQIM | Transmit Queue Interrupt Source Select<br>0: When the transmit queue becomes empty upon completion of message transmission, a transmit queue interrupt request is generated.<br>1: A transmit queue interrupt request is generated each time a message has been transmitted. |
| 12 | TXQIE | Transmit Queue Interrupt Enable<br>0: Transmit queue interrupt is disabled.<br>1: Transmit queue interrupt is enabled. |
| 11 to 8 | TXQDC[3:0] | Transmit Queue Depth Configuration<br>Setting these bits to g (g = 2 to 15) makes the (g + 1)-buffer transmit queue available.<br>Setting these bits to 0 disables the transmit queue.<br>Setting these bits to 1 is prohibited.<br>For transmit buffer merge mode, set g to 2 to 9. |
| 7 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | TXQE | Transmit Queue Enable<br>0: The transmit queue is not used.<br>1: The transmit queue is used. |

#### TXQIM Bit

This bit is used to select a transmit queue interrupt source. Modify this bit in channel reset mode.

#### TXQIE Bit

When the TXQIE bit is set to 1 and the source selected by the TXQIM bit occurs, an interrupt request is generated.

Set the TXQE bit to 0 before modifying the TXQIE bit.

#### TXQDC[3:0] Bits

These bits are used to specify the number of transmit buffers to be allocated to the transmit queues. Transmit buffers are allocated to transmit queues in descending order of buffer number, that is, from (m × 16 + 15) to (m × 16 + 0). For examples of how buffer allocation is done, see **Figure 21.9**.

When the TMME bit in the RSCFDnCFDCmFDCFG register is 1 (transmit buffer merge mode), transmit buffers (m × 16 + 5) to (m × 16 + 0) are merged and cannot be allocated to the transmit queue. Therefore, do not set TXQDC[3:0] bits to 10 to 15.

Modify these bits only in channel reset mode.

#### TXQE Bit

Setting this bit to 1 makes the transmit queue available. Modify this bit in channel communication mode or channel halt mode. This bit is cleared to 0 in channel reset mode.

Before setting the TXQE bit to 1, set the TXQDC[3:0] bits to $0010_B$ or more.

### 21.5.13.2 RSCFDnCFDTXQSTSm — Transmit Queue Status Register (m = 0 to 5)

**Access:** RSCFDnCFDTXQSTSm register can be read or written in 32-bit units
RSCFDnCFDTXQSTSmL register can be read or written in 16-bit units
RSCFDnCFDTXQSTSmLL register can be read or written in 8-bit units

**Address:** RSCFDnCFDTXQSTSm: <RSCANn_base> + $03C0_H$ + ($04_H$ × m)

RSCFDnCFDTXQSTSmL: <RSCANn_base> + $03C0_H$ + ($04_H$ × m)

RSCFDnCFDTXQSTSmLL: <RSCANn_base> + $03C0_H$ + ($04_H$ × m)

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TXQIF | TXQFLL | TXQEMP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W*1 | R | R |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.167 RSCFDnCFDTXQSTSm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 to 8 | Reserved | When read, the undefined value is returned. When writing, write the value after reset. |
| 7 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 | TXQIF | Transmit Queue Interrupt Request Flag<br>0: No transmit queue interrupt request is present.<br>1: A transmit queue interrupt request is present. |
| 1 | TXQFLL | Transmit Queue Full Status Flag<br>0: The transmit queue is not full.<br>1: The transmit queue is full. |
| 0 | TXQEMP | Transmit Queue Empty Status Flag<br>0: The transmit queue contains messages.<br>1: The transmit queue contains no message (transmit queue empty). |

#### TXQIF Flag

The TXQIF flag is set to 1 when the event specified by the TXQIM bit in the RSCFDnCFDTXQCCm register has occurred.

The TXQIF flag is cleared to 0 in channel reset mode or by writing 0 to this flag. This flag is not cleared to 0 by setting the TXQE bit in the RSCFDnCFDTXQCCm register to 0 (the transmit queue is not used).

### TXQFLL Flag

The TXQFLL flag is set to 1 when the number of messages set for the transmit queue matches the transmit queue depth set by the TXQDC[3:0] bits in the RSCFDnCFDTXQCCm register.

This flag is cleared to 0 in any of the following cases.

- The number of messages set for the transmit queue is smaller than the transmit queue depth set by the TXQDC[3:0] bits.
- In channel reset mode

### TXQEMP Flag

The TXQEMP flag is cleared to 0 when even a single message is set for the transmit queue.

This flag is set to 1 in any of the following cases.

- The TXQE bit is set to 0 (the transmit queue is not used).
- The transmit queue becomes empty.
- In channel reset mode

### 21.5.13.3 RSCFDnCFDTXQPCTRm — Transmit Queue Pointer Control Register (m = 0 to 5)

**Access:** RSCFDnCFDTXQPCTRm register is a write-only register that can be written in 32-bit units
RSCFDnCFDTXQPCTRmL register is a write-only register that can be written in 16-bit units
RSCFDnCFDTXQPCTRmLL register is a write-only register that can be written in 8-bit units

**Address:** RSCFDnCFDTXQPCTRm: <RSCANn_base> + $03E0_H$ + ($04_H$ × m)

RSCFDnCFDTXQPCTRmL: <RSCANn_base> + $03E0_H$ + ($04_H$ × m)

RSCFDnCFDTXQPCTRmLL: <RSCANn_base> + $03E0_H$ + ($04_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TXQPC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**Table 21.168 RSCFDnCFDTXQPCTRm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing, write the value after reset. |
| 7 to 0 | TXQPC[7:0] | Transmit Queue Pointer Control<br>Writing $FF_H$ to these bits moves the write pointer of the transmit queue to the next queue buffer. |

#### TXQPC[7:0] Bits

Writing $FF_H$ to the TXQPC[7:0] bits moves the write pointer to the next transmit queue buffer and generates a transmit request of the message. Write transmit messages to the RSCFDnCFDTMIDp, RSCFDnCFDTMPTRp, RSCFDnCFDTMFDCTRp, and RSCFDnCFDTMDFb_p registers (p = 15, 31, 47, 63, 79, and 95) before writing $FF_H$ to the TXQPC[7:0] bits.

When writing $FF_H$ to these bits, make sure that the TXQE bit in the RSCFDnCFDTXQCCm register is set to 1 (the transmit queue is used) and the TXQFLL flag in the RSCFDnCFDTXQSTSm register is 0 (the transmit queue is not full).

## 21.5.14 Details of Transmit History-related Registers

### 21.5.14.1 RSCFDnCFDTHLCCm — Transmit History Configuration and Control Register (m = 0 to 5)

**Access:** RSCFDnCFDTHLCCm register can be read or written in 32-bit units
RSCFDnCFDTHLCCmL register can be read or written in 16-bit units
RSCFDnCFDTHLCCmLL, RSCFDnCFDTHLCCmLH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDTHLCCm: <RSCANn_base> + $0400_H$ + ($04_H$ × m)

RSCFDnCFDTHLCCmL: <RSCANn_base> + $0400_H$ + ($04_H$ × m)

RSCFDnCFDTHLCCmLL: <RSCANn_base> + $0400_H$ + ($04_H$ × m),
RSCFDnCFDTHLCCmLH: <RSCANn_base> + $0401_H$ + ($04_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | THLDTE | THLIM | THLIE | — | — | — | — | — | — | — | THLE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W |

**Table 21.169 RSCFDnCFDTHLCCm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 10 | THLDTE | Transmit History Target Buffer Select<br>0: Entry from transmit/receive FIFO buffers and transmit queue<br>1: Entry from transmit buffers, transmit/receive FIFO buffers, and transmit queue |
| 9 | THLIM | Transmit History Interrupt Source Select<br>0: When 12 sets of data have been stored in the transmit history buffer<br>1: When a single set of transmit history data has been stored |
| 8 | THLIE | Transmit History Interrupt Enable<br>0: Transmit history interrupt is disabled.<br>1: Transmit history interrupt is enabled. |
| 7 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | THLE | Transmit History Buffer Enable<br>0: Transmit history buffer is not used.<br>1: Transmit history buffer is used. |

#### THLDTE Bit

When this bit is set to 0, the transmit history data of messages transmitted from transmit/receive FIFO buffers and the transmit queue is stored in the transmit history buffer. When this bit is set to 1, the transmit history data of messages transmitted from transmit buffers, transmit/receive FIFO buffers, and the transmit queue is stored in the transmit history buffer.

Modify this bit only in channel reset mode.

#### THLIM Bit

This bit is used to select a transmit history interrupt source.

Modify this bit only in channel reset mode.

#### THLIE Bit

When the THLIE bit is set to 1 and the source selected by the THLIM bit has occurred, a transmit history interrupt request is generated. Modify the THLIE bit only when the THLE bit set to 0.

#### THLE Bit

Setting this bit to 1 makes the transmit history buffer available. When data transmission from the buffer selected by the THLDTE bit has been completed, the transmit history data of transmit messages is stored in the transmit history buffer.

Modify this bit in channel communication mode or channel halt mode.

This bit is cleared to 0 in channel reset mode.

### 21.5.14.2 RSCFDnCFDTHLSTSm — Transmit History Status Register (m = 0 to 5)

**Access:** RSCFDnCFDTHLSTSm register can be read or written in 32-bit units
RSCFDnCFDTHLSTSmL register can be read or written in 16-bit units
RSCFDnCFDTHLSTSmLL register can be read or written in 8-bit units
RSCFDnCFDTHLSTSmLH register is a read-only register that can be read in 8-bit units

**Address:** RSCFDnCFDTHLSTSm: <RSCANn_base> + $0420_H$ + ($04_H$ × m)

RSCFDnCFDTHLSTSmL: <RSCANn_base> + $0420_H$ + ($04_H$ × m)

RSCFDnCFDTHLSTSmLL: <RSCANn_base> + $0420_H$ + ($04_H$ × m),
RSCFDnCFDTHLSTSmLH: <RSCANn_base> + $0421_H$ + ($04_H$ × m)

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | THLMC[4:0] | | | | | — | — | — | — | THLIF | THLELT | THLFLL | THLEMP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W*1 | R/W*1 | R | R |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.170 RSCFDnCFDTHLSTSm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 to 8 | THLMC[4:0] | Transmit History Buffer Unread Data Counter<br>These bits indicate the number of unread data sets stored in the transmit history buffer. |
| 7 to 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 | THLIF | Transmit History Interrupt Request Flag<br>0: No transmit history interrupt request is present.<br>1: A transmit history interrupt request is present. |
| 2 | THLELT | Transmit History Buffer Overflow Flag<br>0: Transmit history buffer overflow has not occurred.<br>1: Transmit history buffer overflow has occurred. |
| 1 | THLFLL | Transmit history Buffer Full Status Flag<br>0: Transmit history buffer is not full.<br>1: Transmit history buffer is full. |
| 0 | THLEMP | Transmit History Buffer Empty Status Flag<br>0: Transmit history buffer contains unread data.<br>1: Transmit history buffer contains no unread data (buffer empty). |

#### THLMC[4:0] Bits

These bits indicate the number of unread data sets stored in the transmit history buffer.

These bits are cleared to 0 in channel reset mode.

### THLIF Flag

The THLIF flag is set to 1 when the interrupt source specified with the THLIM bit in the RSCFDnCFDTHLCCm register occurs.

This flag is cleared to 0 in channel reset mode or by the program writing 0 to this flag.

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared.

When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

### THLELT Flag

The THLELT flag is set to 1 when an attempt is made to store new transmit history data while the transmit history buffer is full. In this case, the new data is discarded. This flag becomes 0 in channel reset mode or by the program writing 0 to this flag.

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared.

When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

### THLFLL Flag

The THLFLL flag is set to 1 when 16 data sets have been stored in the transmit history buffer, and is cleared to 0 when the number of data sets stored in the transmit history buffer has decreased to less than 16. This bit is also cleared to 0 in channel reset mode or when the THLE bit in the RSCFDnCFDTHLCCm register is set to 0 (transmit history buffer is not used).

### THLEMP Flag

The THLEMP flag is cleared to 0 when even a single set of transmit history data has been stored in the transmit history buffer.

This flag is set to 1 when all the data in the transmit history buffer has been read. This flag is also set to 1 in channel reset mode or when the THLE bit in the RSCFDnCFDTHLCCm register is set to 0 (transmit history buffer is not used).

**NOTE**

To clear THLIF or THLELT flag to 0, the program must write 0. When writing, use a store instruction to write “0” to the given flag and “1” to other flags.

### 21.5.14.3 RSCFDnCFDTHLPCTRm — Transmit History Pointer Control Register (m = 0 to 5)

**Access:** RSCFDnCFDTHLPCTRm register is a write-only register that can be written in 32-bit units
RSCFDnCFDTHLPCTRmL register is a write-only register that can be written in 16-bit units
RSCFDnCFDTHLPCTRmLL register is a write-only register that can be written in 8-bit units

**Address:** RSCFDnCFDTHLPCTRm: <RSCANn_base> + $0440_H$ + ($04_H$ × m)

RSCFDnCFDTHLPCTRmL: <RSCANn_base> + $0440_H$ + ($04_H$ × m)

RSCFDnCFDTHLPCTRmLL: <RSCANn_base> + $0440_H$ + ($04_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | THLPC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**Table 21.171 RSCFDnCFDTHLPCTRm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing, write the value after reset. |
| 7 to 0 | THLPC[7:0] | Transmit History List Pointer Control<br>Writing $FF_H$ to these bits moves the read pointer to the next unread data in the transmit history buffer. |

#### THLPC[7:0] Bits

When the THLPC[7:0] bits are set to $FF_H$, the read pointer moves to the next data in the transmit history buffer. At this time, the THLMC[4:0] (transmit history buffer unread data counter) value in the RSCFDnCFDTHLSTSm register is decremented. Write $FF_H$ to the THLPC[7:0] bits after reading from the RSCFDnCFDTHLACCm register.

When writing $FF_H$ to these bits, make sure that the THLE bit in the RSCFDnCFDTHLCCm register is set to 1 (transmit history buffer is used) and the THLEMP flag in the RSCFDnCFDTHLSTSm register is 0.

### 21.5.14.4 RSCFDnCFDTHLACCm — Transmit History Access Register (m = 0 to 5)

**Access:** RSCFDnCFDTHLACCm register is a read-only register that can be read in 32-bit units
RSCFDnCFDTHLACCmL, RSCFDnCFDTHLACCmH registers are read-only registers that can be read in 16-bit units
RSCFDnCFDTHLACCmLL, RSCFDnCFDTHLACCmLH, RSCFDnCFDTHLACCmHL, RSCFDnCFDTHLACCmHH registers are read-only registers that can be read in 8-bit units

**Address:** RSCFDnCFDTHLACCm: <RSCANn_base> + $6000_H$ + ($04_H$ × m)

RSCFDnCFDTHLACCmL: <RSCANn_base> + $6000_H$ + ($04_H$ × m),
RSCFDnCFDTHLACCmH: <RSCANn_base> + $6002_H$ + ($04_H$ × m)

RSCFDnCFDTHLACCmLL: <RSCANn_base> + $6000_H$ + ($04_H$ × m),
RSCFDnCFDTHLACCmLH: <RSCANn_base> + $6001_H$ + ($04_H$ × m),
RSCFDnCFDTHLACCmHL: <RSCANn_base> + $6002_H$ + ($04_H$ × m),
RSCFDnCFDTHLACCmHH: <RSCANn_base> + $6003_H$ + ($04_H$ × m)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TID[7:0] | | | | | | | | — | BN[3:0] | | | | BT[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.172 RSCFDnCFDTHLACCm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | TMTS[15:0] | Timestamp Data<br>The timestamp data of stored data can be read. |
| 15 to 8 | TID[7:0] | Label Data<br>The label information of stored data can be read. |
| 7 | Reserved | When read, the value after reset is returned. |
| 6 to 3 | BN[3:0] | Buffer Number Data<br>The buffer number of transmit source (transmit buffer, transmit/receive FIFO or transmit queue) can be read. |
| 2 to 0 | BT[2:0] | Buffer Type Data<br>b2 b1 b0<br>0 0 1: Transmit buffer<br>0 1 0: Transmit/receive FIFO buffer<br>1 0 0: Transmit queue |

#### TMTS[15:0] Bits

Timestamp values in transmit history data stored in the transmit history buffer are displayed.

#### TID[7:0] Bits

These bits indicate the label information of transmit history data stored in the transmit history buffer.

#### BN[3:0] Bits

These bits indicate the transmit source buffer number in the transmit history data stored in the transmit history buffer.

**BT[2:0] Bits**

These bits indicate the type of the transmit source buffer in the transmit history data stored in the transmit history buffer.

## 21.5.15 Details of Test-related Registers

### 21.5.15.1 RSCFDnCFDGTSTCFG — Global Test Configuration Register

**Access:** RSCFDnCFDGTSTCFG register can be read or written in 32-bit units
RSCFDnCFDGTSTCFGL, RSCFDnCFDGTSTCFGH registers can be read or written in 16-bit units
RSCFDnCFDGTSTCFGLL, RSCFDnCFDGTSTCFGHL registers can be read or written in 8-bit units

**Address:** RSCFDnCFDGTSTCFG: <RSCANn_base> + $0468_H$

RSCFDnCFDGTSTCFGL: <RSCANn_base> + $0468_H$,
RSCFDnCFDGTSTCFGH: <RSCANn_base> + $046A_H$

RSCFDnCFDGTSTCFGLL: <RSCANn_base> + $0468_H$,
RSCFDnCFDGTSTCFGHL: <RSCANn_base> + $046A_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | RTMPS[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | C5ICBCE | C4ICBCE | C3ICBCE | C2ICBCE | C1ICBCE | C0ICBCE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.173 RSCFDnCFDGTSTCFG Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 22 to 16 | RTMPS[6:0] | RAM Test Page Configuration<br>Set a value within the range of page 0 ($00_H$) to page 83 ($53_H$). |
| 15 to 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 | C5ICBCE | CAN5 Inter-channel Communication Test Enable<br>0: CAN5 inter-channel communication test is disabled<br>1: CAN5 inter-channel communication test is enabled. |
| 4 | C4ICBCE | CAN4 Inter-channel Communication Test Enable<br>0: CAN4 inter-channel communication test is disabled<br>1: CAN4 inter-channel communication test is enabled. |
| 3 | C3ICBCE | CAN3 Inter-channel Communication Test Enable<br>0: CAN3 inter-channel communication test is disabled<br>1: CAN3 inter-channel communication test is enabled. |
| 2 | C2ICBCE | CAN2 Inter-channel Communication Test Enable<br>0: CAN2 inter-channel communication test is disabled<br>1: CAN2 inter-channel communication test is enabled. |
| 1 | C1ICBCE | CAN1 Inter-Channel Communication Test Enable<br>0: CAN1 inter-channel communication test is disabled.<br>1: CAN1 inter-channel communication test is enabled. |
| 0 | C0ICBCE | CAN0 Inter-Channel Communication Test Enable<br>0: CAN0 inter-channel communication test is disabled.<br>1: CAN0 inter-channel communication test is enabled. |

Modify the RSCFDnCFDGTSTCFG register only in global test mode.

#### RTMPS[6:0] Bits

These bits are used to set the RAM test target page number for RAM test. Set a value in the range of $00_H$ to $53_H$, inclusive.

In CAN FD mode, do not access more than 64 bytes in the last page (RTMPS = $53_H$) during RAM test.

#### C5ICBCE Bit

Setting this bit to 1 enables the channel 5 inter-channel communication test.

This bit is cleared to 0 in global reset mode.

#### C4ICBCE Bit

Setting this bit to 1 enables the channel 4 inter-channel communication test.

This bit is cleared to 0 in global reset mode.

#### C3ICBCE Bit

Setting this bit to 1 enables the channel 3 inter-channel communication test.

This bit is cleared to 0 in global reset mode.

#### C2ICBCE Bit

Setting this bit to 1 enables the channel 2 inter-channel communication test.

This bit is cleared to 0 in global reset mode.

#### C1ICBCE Bit

Setting this bit to 1 enables the channel 1 inter-channel communication test.

This bit is cleared to 0 in global reset mode.

#### C0ICBCE Bit

Setting this bit to 1 enables the channel 0 inter-channel communication test.

This bit is cleared to 0 in global reset mode.

### 21.5.15.2 RSCFDnCFDGTSTCTR — Global Test Control Register

**Access:** RSCFDnCFDGTSTCTR register can be read or written in 32-bit units
RSCFDnCFDGTSTCTRL register can be read or written in 16-bit units
RSCFDnCFDGTSTCTRLL register can be read or written in 8-bit units

**Address:** RSCFDnCFDGTSTCTR: <RSCANn_base> + $046C_H$

RSCFDnCFDGTSTCTRL: <RSCANn_base> + $046C_H$

RSCFDnCFDGTSTCTRLL: <RSCANn_base> + $046C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | RTME | — | ICBCTME |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R/W |

**Table 21.174 RSCFDnCFDGTSTCTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 | RTME | RAM Test Enable<br>0: RAM test is disabled.<br>1: RAM test is enabled. |
| 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | ICBCTME | Communication Test between Channels Enable<br>0: Communication test between channels disabled<br>1: Communication test between channels enabled |

#### RTME Bit

Setting this bit to 1 enables the RAM test. Modify this bit only in global test mode.

This bit is cleared to 0 in global reset mode.

1. Set the GMDC[1:0] bits in the RSCFDnCFDGCTR register to $10_B$ (Global test mode).
2. Set the RTME bit to 1.
3. Check that the RTME bit is set to 1.

#### ICBCTME Bit

When this bit is set to 1, a communication test is enabled between the channels for which the CmICBCE bit (m = 0 to 5) in the RSCFDnCFDGTSTCFG register has been set to 1. Modify the ICBCTME bit only in global test mode.

This bit is cleared to 0 in global reset mode.

### 21.5.15.3 RSCFDnCFDGLOCKK — Global Lock Key Register

**Access:** RSCFDnCFDGLOCKK register is a write-only register that can be written in 32-bit units.
RSCFDnCFDGLOCKKL register is a write-only register that can be written in 16-bit units.

**Address:** RSCFDnCFDGLOCKK: <RSCANn_base> + $047C_H$

RSCFDnCFDGLOCKKL: <RSCANn_base> + $047C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LOCK[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 |

Note 1. Writing to these bits is effective only when the RS-CANFD module is in global test mode.

**Table 21.175 RSCFDnCFDGLOCKK Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When writing, write the value after reset. |
| 15 to 0 | LOCK[15:0] | Lock Key<br>These bits are key bits to release protection of test mode. |

The RSCFDnCFDGLOCKK register is used to release the protection of special test bits and is write only.

For the protection release data, see **Section 21.11.4.2, Procedure for Releasing the Protection**.

#### LOCK[15:0] Bits

Writing the protection release data to the LOCK[15:0] bits in succession enables writing 1 to the RTME bit in the RSCFDnCFDGTSTCTR register.

After the protection has been released, writing to the I/O register area (<RSCANn_base> + $0000_H$ to <RSCANn_base> + $0FFF_H$) of the CAN (except the RAM) enables the protection again.

Reading from the I/O register area of the CAN or reading from/writing to other areas does not enable the protection.

### 21.5.15.4 RSCFDnCFDRPGACCr — RAM Test Page Access Register (r = 0 to 63)

**Access:** RSCFDnCFDRPGACCr register can be read or written in 32-bit units
RSCFDnCFDRPGACCrL, RSCFDnCFDRPGACCrH registers can be read or written in 16-bit units
RSCFDnCFDRPGACCrLL, RSCFDnCFDRPGACCrLH, RSCFDnCFDRPGACCrHL,
RSCFDnCFDRPGACCrHH registers can be read or written in 8-bit units

**Address:** RSCFDnCFDRPGACCr: <RSCANn_base> + $6400_H$ + ($04_H$ × r)

RSCFDnCFDRPGACCrL: <RSCANn_base> + $6400_H$ + ($04_H$ × r),
RSCFDnCFDRPGACCrH: <RSCANn_base> + $6402_H$ + ($04_H$ × r)

RSCFDnCFDRPGACCrLL: <RSCANn_base> + $6400_H$ + ($04_H$ × r),
RSCFDnCFDRPGACCrLH: <RSCANn_base> + $6401_H$ + ($04_H$ × r),
RSCFDnCFDRPGACCrHL: <RSCANn_base> + $6402_H$ + ($04_H$ × r),
RSCFDnCFDRPGACCrHH: <RSCANn_base> + $6403_H$ + ($04_H$ × r)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDTA[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDTA[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.176 RSCFDnCFDRPGACCr Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | RDTA[31:0] | RAM Data Test Access<br>Data can be read and written in RSCAN RAM. |

Modify the RSCFDnCFDRPGACCr register in global test mode with the RTME bit in the RSCFDnCFDGTSTCTR register set to 1 (RAM test is enabled).

The RSCFDnCFDRPGACCr register is readable and writable when the RTME bit is set to 1.

## 21.6 Interrupt Sources and DMA Trigger

### 21.6.1 Interrupt Sources

The RS-CANFD module has 20 interrupts that are grouped into global interrupts and channel interrupts.

- Global interrupts (2 sources):
  Receive FIFO interrupt
  Global error interrupt
- Channel interrupts (3 sources/channel):
  CANm transmit interrupt (m = 0 to 5)
  - CANm transmit complete interrupt
  - CANm transmit abort interrupt
  - CANm transmit/receive FIFO transmit complete interrupt (in transmit mode, gateway mode)
  - CANm transmit history interrupt
  - CANm transmit queue Interrupt

  CANm transmit/receive FIFO receive complete interrupt (in receive mode, gateway mode)
  CANm error interrupt

When an interrupt request is generated, the corresponding interrupt request flag is set to 1 (interrupt request present). In that case, when the interrupt enable bit is set to 1 (enabling interrupts), an interrupt request is output from the RS-CANFD module. (Generation of interrupts also depends on the interrupt control register settings of the interrupt controller.)

Setting the interrupt request flag to 0 (no interrupt request present) or setting the interrupt enable bit to 0 (disabling interrupts) clears the current interrupt request. The current interrupt request is still output until the interrupt request flag is cleared.

**Table 21.177** lists the CAN interrupt sources. **Figure 21.2** shows the CAN global interrupt block diagram. **Figure 21.3** shows the CAN channel interrupt block diagram.

**Table 21.177 List of CAN Interrupt Sources**

| Interrupt Source | | | Corresponding Interrupt Request Flag | Corresponding Interrupt Enable Bit |
|---|---|---|---|---|
| Global interrupts | Receive FIFO | Receive FIFO 0 | RFIF in the RSCANn(CFD)RFSTS0 register | RFIE in the RSCANn(CFD)RFCC0 register |
| | | Receive FIFO 1 | RFIF in the RSCANn(CFD)RFSTS1 register | RFIE in the RSCANn(CFD)RFCC1 register |
| | | Receive FIFO 2 | RFIF in the RSCANn(CFD)RFSTS2 register | RFIE in the RSCANn(CFD)RFCC2 register |
| | | Receive FIFO 3 | RFIF in the RSCANn(CFD)RFSTS3 register | RFIE in the RSCANn(CFD)RFCC3 register |
| | | Receive FIFO 4 | RFIF in the RSCANn(CFD)RFSTS4 register | RFIE in the RSCANn(CFD)RFCC4 register |
| | | Receive FIFO 5 | RFIF in the RSCANn(CFD)RFSTS5 register | RFIE in the RSCANn(CFD)RFCC5 register |
| | | Receive FIFO 6 | RFIF in the RSCANn(CFD)RFSTS6 register | RFIE in the RSCANn(CFD)RFCC6 register |
| | | Receive FIFO 7 | RFIF in the RSCANn(CFD)RFSTS7 register | RFIE in the RSCANn(CFD)RFCC7 register |
| | Global error | | • DEF in the RSCANn(CFD)GERFL register<br>• MES in the RSCANn(CFD)GERFL register<br>• THLES in the RSCANn(CFD)GERFL register<br>• CMPOF in the RSCFDnCFDGERFL register | • DEIE in the RSCANn(CFD)GCTR register<br>• MEIE in the RSCANn(CFD)GCTR register<br>• THLEIE in the RSCANn(CFD)GCTR register<br>• CMPOFIE in the RSCFDnCFDGCTR register |
| Channel interrupts (m = 0 to 5) | CANm transmit | CANm transmit complete | TMTRF[1:0] in the RSCANn(CFD)TMSTSp register | TMIEp in the RSCANn(CFD)TMIECy register |
| | | CANm transmit abort | TMTRF[1:0] in the RSCANn(CFD)TMSTSp register | TAIE in the RSCANn(CFD)CmCTR register |
| | | CANm transmit/ receive FIFO transmit complete | CFTXIF in the RSCANn(CFD)CFSTSk register | CFTXIE in the RSCANn(CFD)CFCCk register |
| | | CANm transmit queue | TXQIF in the RSCANn(CFD)TXQSTSm register | TXQIE in the RSCANn(CFD)TXQCCm register |
| | | CANm transmit history | THLIF in the RSCANn(CFD)THLSTSm register | THLIE in the RSCANn(CFD)THLCCm register |
| | CANm transmit/receive FIFO receive complete | | CFRXIF in the RSCANn(CFD)CFSTSk register | CFRXIE in the RSCANn(CFD)CFCCk register |
| | CANm error | | • BEF in the RSCANn(CFD)CmERFL register<br>• ALF in the RSCANn(CFD)CmERFL register<br>• BLF in the RSCANn(CFD)CmERFL register<br>• OVLF in the RSCANn(CFD)CmERFL register<br>• BORF in the RSCANn(CFD)CmERFL register<br>• BOEF in the RSCANn(CFD)CmERFL register<br>• EPF in the RSCANn(CFD)CmERFL register<br>• EWF in the RSCANn(CFD)CmERFL register<br>• SOCO in the RSCFDnCFDCmFDSTS register<br>• EOCO in the RSCFDnCFDCmFDSTS register<br>• TDCVF in the RSCFDnCFDCmFDSTS register | • BEIE in the RSCANn(CFD)CmCTR register<br>• ALIE in the RSCANn(CFD)CmCTR register<br>• BLIE in the RSCANn(CFD)CmCTR register<br>• OLIE in the RSCANn(CFD)CmCTR register<br>• BORIE in the RSCANn(CFD)CmCTR register<br>• BOEIE in the RSCANn(CFD)CmCTR register<br>• EPIE in the RSCANn(CFD)CmCTR register<br>• EWIE in the RSCANn(CFD)CmCTR register<br>• SOCOIE in the RSCFDnCFDCmCTR register<br>• EOCOIE in the RSCFDnCFDCmCTR register<br>• TDCVFIE in the RSCFDnCFDCmCTR register |

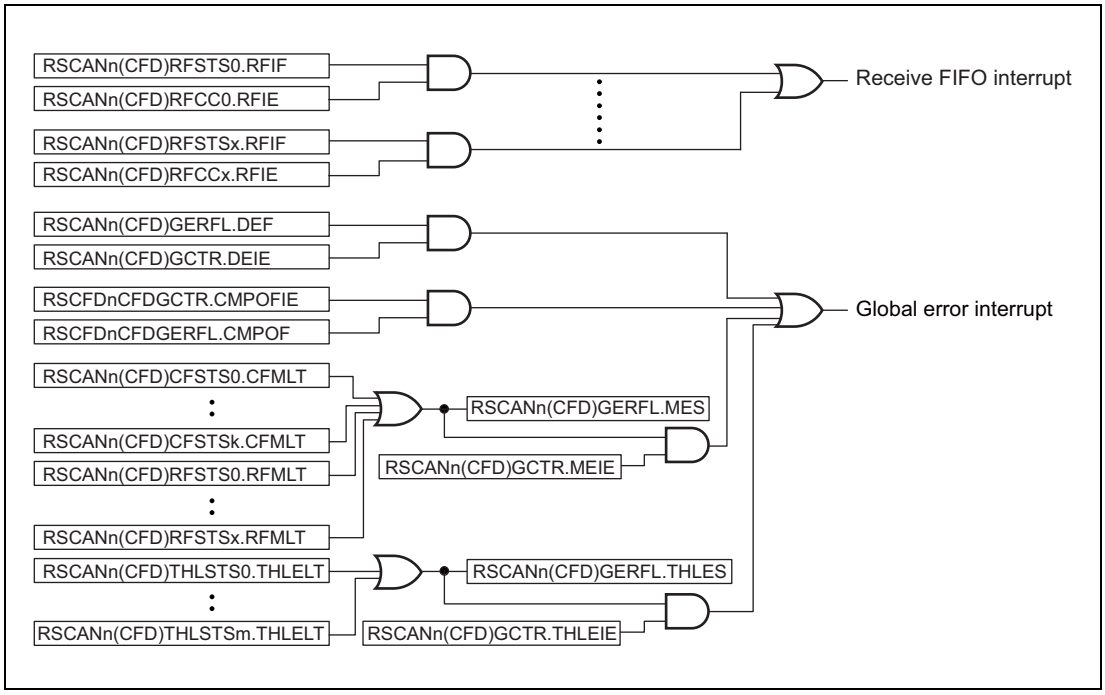



**Figure 21.2 CAN Global Interrupt Block Diagram**

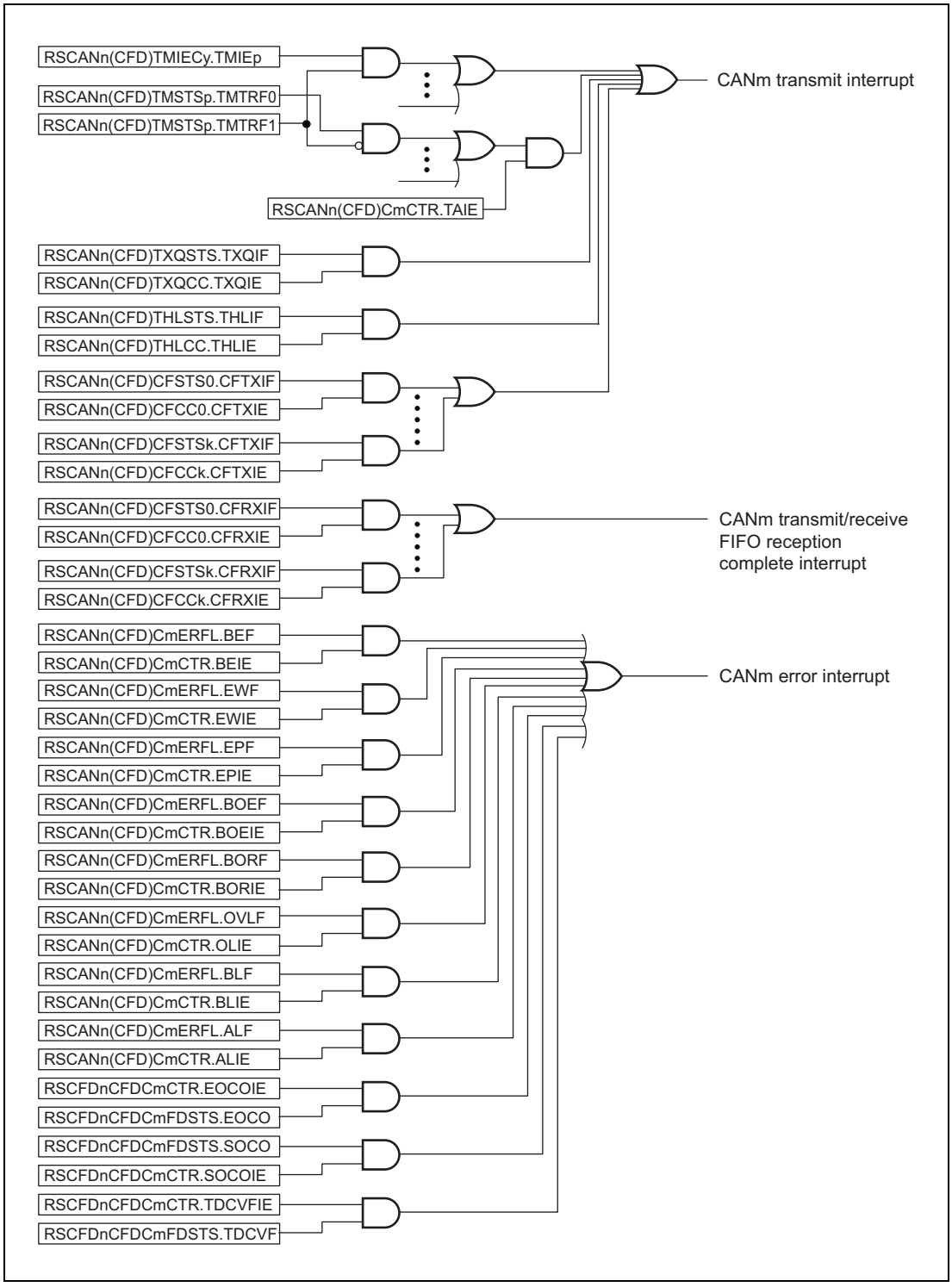



**Figure 21.3 CAN Channel Interrupt Block Diagram**

### 21.6.2 DMA Trigger (Only in CAN FD Mode)

In CAN FD mode, receive FIFO buffers can be related to DMA channels. The following 14 FIFO buffers can be related.

- All receive FIFO buffers x (x = 0 to 7)
- The first transmit/receive FIFO buffer k (k = 3 × m, m = 0 to 5) allocated to channel m

When the DMA enable bit (RFDMAEx or CFDMAEm bit in the RSCFDnCFDCDTCT register) is set to 1 and an unread message is remaining in the related FIFO, a DMA transfer request trigger is generated.

## 21.7 CAN Modes

The RS-CANFD module has four global modes to control the entire RS-CANFD module status and four channel modes to control individual channel status. Details of global modes are described in **Section 21.7.1, Global Modes**, and details of channel modes are described in **Section 21.7.2, Channel Modes**.

- Global stop mode: Stops the clocks of the entire module to achieve low power consumption.
- Global reset mode: Performs initial settings for the entire module.
- Global test mode: Performs test settings and performs the RAM test.
- Global operating mode: Makes the entire module operable.
- Channel stop mode: Stops the channel clock.
- Channel reset mode: Performs initial settings for the channels.
- Channel halt mode: Stops CAN communication and allows channel testing.
- Channel communication mode: Performs CAN communication.

### 21.7.1 Global Modes

**Figure 21.4** shows the transitions of global modes.

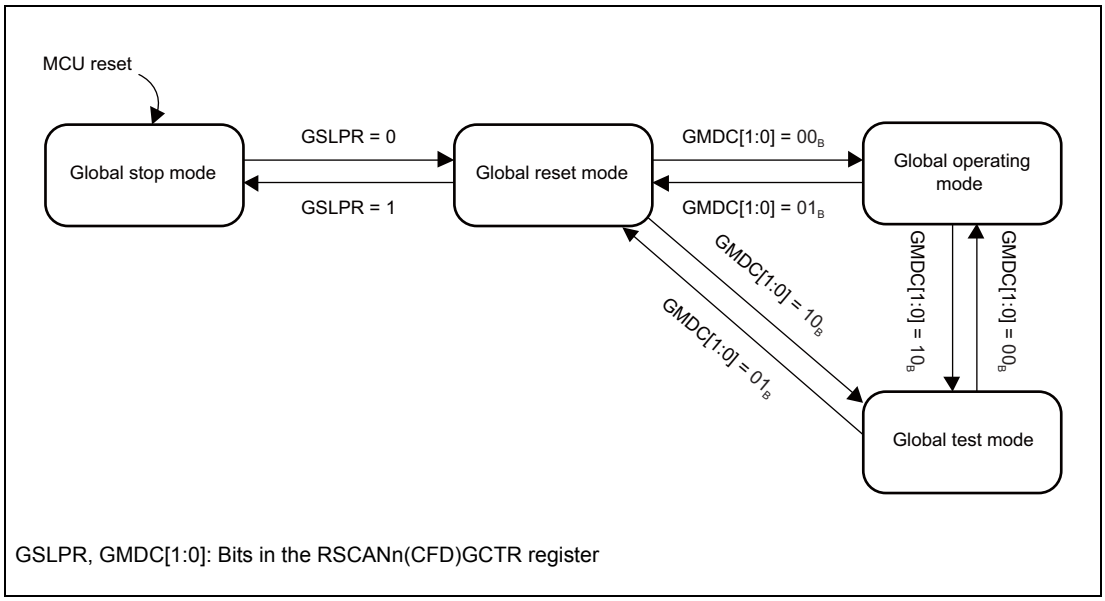



**Figure 21.4 Transitions of Global Modes**

In some cases, global mode transitions also force channel mode transitions. **Table 21.178** shows the channel mode transitions depending on the global mode setting dictated by the GMDC[1:0] bits and the GSLPR bit.

**Table 21.178 Transitions of Channel Modes Depending on Global Mode Setting (GMDC[1:0] and GSLPR Bits)**

| Channel Mode before Setting | Channel Mode after Setting | | | |
|---|---|---|---|---|
| | **GMDC[1:0] = $00_B$ GSLPR = 0 (Global Operation)** | **GMDC[1:0] = $10_B$ GSLPR = 0 (Global Test)** | **GMDC[1:0] = $01_B$ GSLPR = 0 (Global Reset)** | **GMDC[1:0] = $01_B$ GSLPR = 1 (Global Stop)** |
| Channel communication | Channel communication | Channel halt | Channel reset | Transition prohibited |
| Channel halt | Channel halt | Channel halt | Channel reset | Transition prohibited |
| Channel reset | Channel reset | Channel reset | Channel reset | Channel stop |
| Channel stop | Channel stop | Channel stop | Channel stop | Channel stop |

**Note:** GMDC[1:0], GSLPR: Bits in the RSCANn(CFD)GCTR register

**Table 21.179** shows the global mode transition time.

**Table 21.179 Global Mode Transition Time**

| Mode before Transition | Mode after Transition | Maximum Transition Time |
|---|---|---|
| Global stop | Global reset | Three pclk cycles |
| Global reset | Global stop | Three pclk cycles |
| Global reset | Global test | Ten pclk cycles |
| Global reset | Global operating | Ten pclk cycles |
| Global test | Global reset | Two CAN bit times*1,*2 |
| Global test | Global operating | Three pclk cycles |
| Global operating | Global reset | Two CAN bit times*1,*2 |
| Global operating | Global test | Two CAN frames*1 |

Note 1. CAN frame time and CAN bit time of the lowest communication speed of the channels in use

Note 2. In CAN FD mode, this time value is the CAN bit time of the nominal bit rate.

#### 21.7.1.1 Global Stop Mode

In global stop mode, clocks of the CAN do not run and therefore power consumption is reduced. CAN registers can be read, but writing data to them is prohibited. Register values are retained. Only the clock used by the CPU for writing to the GSLPR bit runs in this mode.

After the MCU is reset, the RS-CANFD module transitions to global stop mode. Setting the GSLPR bit in the RSCANn(CFD)GCTR register to 1 (global stop mode) in global reset mode sets the CSLPR bit in each of the RSCANn(CFD)CmCTR register to 1 (channel stop mode). Afterwards, if all channels are forced to transition to channel stop mode, the RS-CANFD module transitions to global stop mode. The GSLPR bit should not be modified in global operating mode or global test mode.

#### 21.7.1.2 Global Reset Mode

In global reset mode, RS-CANFD module settings are performed. When the RS-CANFD module transitions to global reset mode, some registers are initialized. For registers to be initialized, see **Table 21.182, Registers Initialized in Global Reset Mode and Channel Reset Mode** and **Table 21.183, Registers Initialized Only in Global Reset Mode**.

Setting the GMDC[1:0] bits in the RSCANn(CFD)GCTR register to $01_B$ sets the CHMDC[1:0] bits in each of the RSCANn(CFD)CmCTR registers (m = 0 to 5) to $01_B$ (channel reset mode). If all channels are forced to transition to channel reset mode, the RS-CANFD module transitions to global reset mode. Channels that are already in channel reset mode or channel stop mode do not transition (because the CHMDC[1:0] bits have already been set to $01_B$).

#### 21.7.1.3 Global Test Mode

In global test mode, settings for test-related registers are performed. When the RS-CANFD module transitions to global test mode, all CAN communications are disabled.

Setting the GMDC[1:0] bits in the RSCANn(CFD)GCTR register to $10_B$ sets the CHMDC[1:0] bits in each of the RSCANn(CFD)CmCTR register to $10_B$ (channel halt mode). If all channels are forced to transition to channel halt mode, the RS-CANFDmodule transitions to global test mode. Channels that are in channel stop mode, channel reset mode, or channel halt mode do not transition.

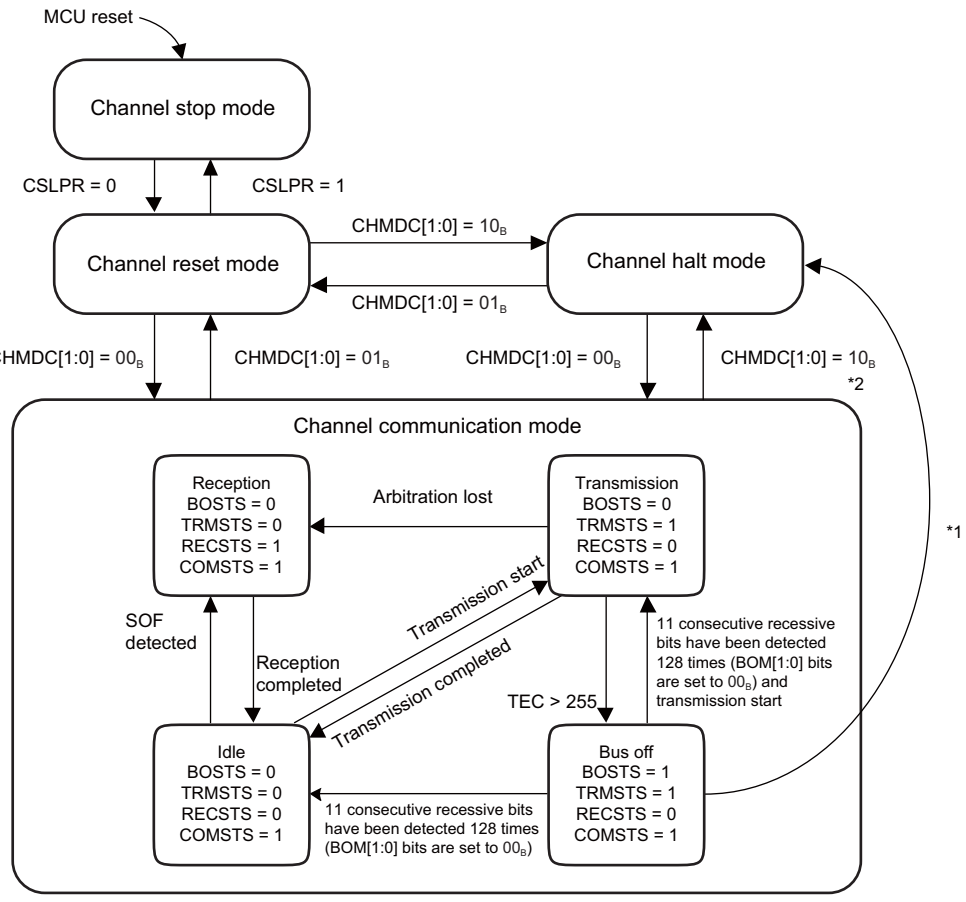
#### 21.7.1.4 Global Operating Mode

The RS-CANFD module operates in global operating mode.

When the GMDC[1:0] bits in the RSCANn(CFD)GCTR register are set to $00_B$, the RS-CANFD module transitions to global operating mode.

## 21.7.2 Channel Modes

**Figure 21.5** shows a channel mode state transition chart. **Table 21.180** shows the channel mode transition time.




**Note:** CHMDC[1:0], CSLPR, BOM[1:0]: Bits in the RSCANn(CFD)CmCTR register (m = 0 to 5)
BOSTS, TRMSTS, RECSTS, COMSTS: Bits in the RSCANn(CFD)CmSTS register

Note 1. Timing of transition from bus off state to channel halt mode
- When BOM[1:0] = $01_B$: Transition to channel halt mode when TEC exceeds 255
- When BOM[1:0] = $10_B$: Transition to channel halt mode when 11 consecutive recessive bits have been detected 128 times
- When BOM[1:0] = $11_B$: Transition to channel halt mode when the CHMDC[1:0] bits are set to $10_B$

Note 2. While the CAN bus is locked at the dominant level, transition to channel halt mode is not made. In that case, enter channel reset mode.

**Figure 21.5 Channel Mode State Transition Chart**

Table 21.180 Channel Mode Transition Time

| Mode before Transition | Mode after Transition | Maximum Transition Time |
|---|---|---|
| Channel stop | Channel reset | Three pclk cycles |
| Channel reset | Channel stop | Three pclk cycles |
| Channel reset | Channel halt | Three CANm bit times*1 |
| Channel reset | Channel communication | Four CANm bit times*1 |
| Channel halt | Channel reset | Two CANm bit times*1 |
| Channel halt | Channel communication | Four CANm bit times*1 |
| Channel communication | Channel reset | Two CANm bit times*1 |
| Channel communication | Channel halt | Two CANm frames |

Note 1. In CAN FD mode, this time value is the CANm bit time of the nominal bit rate.

#### 21.7.2.1 Channel Stop Mode

In channel stop mode, clocks are not supplied to channels and therefore power consumption is reduced. CAN registers can be read, but writing data to them is prohibited. Register values are retained.

Each channel enters channel stop mode after the MCU is reset. Channels also transition to channel stop mode when the CSLPR bit in the RSCANn(CFD)CmCTR register (m = 0 to 5) is set to 1 (channel stop mode) in channel reset mode. The CSLPR bit should not be modified in channel communication mode and channel halt mode.

#### 21.7.2.2 Channel Reset Mode

In channel reset mode, channel settings are performed. When a channel transitions to channel reset mode, some channel-related registers are initialized. For registers to be initialized, see **Table 21.182, Registers Initialized in Global Reset Mode and Channel Reset Mode**.

When the CHMDC[1:0] bits in the RSCANn(CFD)CmCTR register are set to $01_B$ (channel reset mode) during CAN communication, communication is terminated before it is completed and the channel transitions to channel reset mode. **Table 21.181** shows the operation when the CHMDC[1:0] bits are set to $01_B$ (channel reset mode) during CAN communication.

### 21.7.2.3 Channel Halt Mode

In channel halt mode, settings for test-related registers of channels are performed. When a channel transitions to channel halt mode, CAN communication of the channel stops.

**Table 21.181** shows operation when the CHMDC[1:0] bits are set to $10_B$ (channel halt mode) during CAN communication.

**Table 21.181 Operation when a Channel Transitions to Channel Reset Mode/Channel Halt Mode**

| Mode | During Reception | During Transmission | Bus Off State |
|---|---|---|---|
| Channel reset (CHMDC[1:0] = $01_B$) | Transitions to channel reset mode before reception is completed.*1 | Transitions to channel reset mode before transmission is completed.*1 | Transitions to channel reset mode before bus off recovery. |
| Channel halt*3 (CHMDC[1:0] = $10_B$) | Transitions to channel halt mode after reception is completed.*2 | Transitions to channel halt mode after transmission is completed. | [When BOM[1:0] = $00_B$] Transitions to channel halt mode (CHMDC[1:0] = $10_B$) only after bus off recovery. [When BOM[1:0] = $01_B$] Transitions to channel halt mode automatically when the condition for transition to bus off state is met. [When BOM[1:0] = $10_B$] Transitions to channel halt mode automatically after bus off recovery. [When BOM[1:0] = $11_B$] Transitions to channel halt mode immediately after the CHMDC[1:0] bits are set to $10_B$ before bus off recovery. |

Note 1. To allow transition to channel reset mode after communication is completed, set the CHMDC[1:0] bits to $10_B$ and confirm that communication has been completed and transition to channel halt mode has been made, and then set the CHMDC[1:0] bits to $01_B$.

Note 2. While the CAN bus is locked at the dominant level, transition to channel halt mode is not made. In that case, enter channel reset mode. The CAN bus status can be confirmed with the BLF flag of the RSCANn(CFD)CmERFL register that becomes 1 when bus dominant lock is detected.

Note 3. In classical CAN mode, when the transition from channel reset mode to channel halt mode is to be made, set the RSCANnCmCFG register in channel reset mode and then shift to channel halt mode. In CAN FD mode, set the RSCFDnCFDCmNCFG register and the RSCFDnCFDCmDCFG register, and then make a transition.

### 21.7.2.4 Channel Communication Mode

In channel communication mode, CAN communication is performed. Each channel has the following communication states during CAN communication.

- Idle: Neither reception nor transmission is in progress.
- Reception: Receiving a message sent from another node.
- Transmission: Transmitting a message.
- Bus off: Isolated from CAN communication.

When the CHMDC[1:0] bits in the RSCANn(CFD)CmCTR register are set to $00_B$, the channel transitions to channel communication mode. After that, once 11 consecutive recessive bits have been detected, the COMSTS flag in the RSCANn(CFD)CmSTS register (m = 0 to 5) is set to 1 (communication is ready) and transmission and reception are enabled on the CAN network as an active node. At this time, transmission and reception of messages can be started.

#### 21.7.2.5 Bus Off State

A channel transitions to the bus off state according to the transmit/receive error counter increment/ decrement rules of the CAN specifications.

The conditions for returning from the bus off state are determined by the BOM[1:0] bits in the RSCANn(CFD)CmCTR register.

- When BOM[1:0] = $00_B$:
  Bus off recovery is compliant with the CAN specifications. After 11 consecutive recessive bits have been detected 128 times, a channel returns from the bus off state to the CAN communication ready state (error active state). At that time, the TEC[7:0] and REC[7:0] bits in the RSCANn(CFD)CmSTS register are initialized to $00_H$, the BORF flag in the RSCANn(CFD)CmERFL register is set to 1 (bus off recovery is detected), and a bus off recovery interrupt request is generated. When the CHMDC[1:0] bits in the RSCANn(CFD)CmCTR register are set to $10_B$ (channel halt mode) in the bus off state, the channel transitions to channel halt mode after bus off recovery has been completed (11 consecutive recessive bits have been detected 128 times).
- When BOM[1:0] = $01_B$:
  When a channel transitions to the bus off state, the CHMDC[1:0] bits are set to $10_B$ and the channel transitions to channel halt mode. At that time, the TEC[7:0] and REC[7:0] bits are initialized to $00_H$. The BORF flag is not set to 1, and bus off recovery interrupt request is not generated.
- When BOM[1:0] = $10_B$:
  When a channel has transitioned to the bus off state, the CHMDC[1:0] bits are set to $10_B$. After bus off recovery has been completed (11 consecutive recessive bits have been detected 128 times), the channel transitions to channel halt mode. At that time, the TEC[7:0] and REC[7:0] bits are initialized to $00_H$, the BORF flag is set to 1, and a bus off recovery interrupt request is generated.
- When BOM[1:0] = $11_B$:
  When the CHMDC[1:0] bits are set to $10_B$ in the bus off state, the channel transitions to channel halt mode before bus off recovery is completed. At that time, the TEC[7:0] and REC[7:0] bits are initialized to $00_H$, but the BORF flag is not set to 1. Also, a bus off recovery interrupt is not generated.
  However, the BORF flag becomes 1 and a bus off recovery interrupt request is generated if a RS-CANFD module transitions to error active state (by detecting 128 times of 11 consecutive recessive bits) before CHMDC[1:0] bits are set to $10_B$.

If the RS-CANFD module causes the channel to transition to channel halt mode simultaneously with a program write to the CHMDC[1:0] bits, the program write takes precedence. An automatic transition to channel halt mode when the BOM[1:0] bits are set to $01_B$ or $10_B$ is made only when the CHMDC[1:0] bits are $00_B$ (channel communication mode).

Furthermore, setting the RTBO bit in the RSCANn(CFD)CmCTR register to 1 allows a forced return from the bus off state. As soon as the RTBO bit is set to 1, the state changes to the error active state. After 11 consecutive recessive bits have been detected, the RS-CANFD module becomes ready for communication. In this case, the BORF flag is not set to 1 and the TEC[7:0] and REC[7:0] bits are initialized to $00_H$. Write 1 to the RTBO bit only when the BOM[1:0] value is $00_B$. Writing the RTBO bit to 1 in a state other than the bus off state is ignored, and the RTBO bit is immediately set to 0.

### 21.7.3 Initializing Registers by Transition to CAN Mode

**Table 21.182** lists bits and flags to be initialized by a transition to channel reset mode. These bits and flags are also initialized by a transition to global reset mode. Furthermore, **Table 21.183** lists bits and flags to be initialized only by a transition to global reset mode.

**Table 21.182 Registers Initialized in Global Reset Mode and Channel Reset Mode**

| Register | Bit / Flag |
|---|---|
| RSCANn(CFD)CmCTR register | (ROM), CRCT, CTMS[1:0], CTME, CHMDC[1:0] |
| RSCANn(CFD)CmSTS register | CHLTSTS, EPSTS, BOSTS, TRMSTS, RECSTS, COMSTS, (ESIF), REC[7:0], TEC[7:0] |
| RSCANn(CFD)CmERFL register | CRCREG[14:0], ADERR, B0ERR, B1ERR, CERR, AERR, FERR, SERR, ALF, BLF, OVLF, BORF, BOEF, EPF, EWF, BEF |
| RSCFDnCFDCmFDCTR register | EOCCLR, SOCCLR |
| RSCFDnCFDCmFDSTS register | SOC[7:0], EOC[7:0], SOCO, EOCO, TDCVF, TDCR[6:0] |
| RSCFDnCFDCmFDCRC register | CRCREG[20:0] |
| RSCANn(CFD)CFCCk register | When transmit/receive FIFO buffer is in transmit mode or gateway mode: CFE |
| RSCANn(CFD)CFSTSk register | When transmit/receive FIFO buffer is in transmit mode or gateway mode: CFMC[7:0], CFFLL, CFEMP, CFMLT, CFRXIF, CFTXIF |
| RSCANn(CFD)CFTISTS register | CFkTXIF |
| RSCANn(CFD)TMCp register | TMOM, TMTAR, TMTR |
| RSCANn(CFD)TMSTSp register | TMTARM, TMTRM, TMTRF[1:0], TMTSTS |
| RSCANn(CFD)TMTRSTSy register | TMTRSTSp (Bits of corresponding channel are initialized in channel reset mode.) |
| RSCANn(CFD)TMTARSTSy register | TMTARSTSp (Bits of corresponding channel are initialized in channel reset mode.) |
| RSCANn(CFD)TMTCSTSy register | TMTCSTSp (Bits of corresponding channel are initialized in channel reset mode.) |
| RSCANn(CFD)TMTASTSy register | TMTASTSp (Bits of corresponding channel are initialized in channel reset mode.) |
| RSCANn(CFD)TXQCCm register | TXQE |
| RSCANn(CFD)TXQSTSm register | TXQIF, TXQFLL, TXQEMP |
| RSCANn(CFD)THLCCm register | THLE |
| RSCANn(CFD)THLSTSm register | THLMC[4:0], THLIF, THLELT, THLFLL, THLEMP |
| RSCANn(CFD)GTINTSTS0 register | TSIFm, TAIFm, TQIFm, CFTIFm, THIFm (m = 0 to 3) |
| RSCANn(CFD)GTINTSTS1 register | TSIFm, TAIFm, TQIFm, CFTIFm, THIFm (m = 4, 5) |

**Note:** Bits and flags in parentheses exist only in registers in CAN FD mode.

**Table 21.183 Registers Initialized Only in Global Reset Mode**

| Register | Bit / Flag |
|---|---|
| RSCANn(CFD)GSTS register | GHLTSTS |
| RSCANn(CFD)GERFL register | EEF0, EEF1, EEF2, EEF3, EEF4, EEF5, (CMPOF), THLES, MES, DEF |
| RSCANn(CFD)GTSC register | TS[15:0] |
| RSCANn(CFD)RMNDy register | RMNSq |
| RSCANn(CFD)RFCCx register | RFE |
| RSCANn(CFD)RFSTSx register | RFMC[7:0], RFIF, RFMLT, RFFLL, RFEMP |
| RSCANn(CFD)CFCCk register | When transmit/receive FIFO buffer is in receive mode: CFE |
| RSCANn(CFD)CFSTSk register | When transmit/receive FIFO buffer is in receive mode: CFMC[7:0], CFFLL, CFEMP, CFTXIF, CFRXIF, CFMLT |
| RSCANn(CFD)FESTS register | CFkEMP, RFxEMP |
| RSCANn(CFD)FFSTS register | CFkFLL, RFxFLL |
| RSCANn(CFD)FMSTS register | CFkMLT, RFxMLT |
| RSCANn(CFD)RFISTS register | RFxIF |
| RSCANn(CFD)CFRISTS register | CFkRXIF |
| RSCFDnCFDCDTCT register | CFDMAEm, RFDMAEx |
| RSCFDnCFDCDTSTS register | CFDMASTSm, RFDMASTSx |
| RSCANn(CFD)GTSTCFG register | RTMPS[6:0], C0ICBCE, C1ICBCE, C2ICBCE, C3ICBCE, C4ICBCE, C5ICBCE |
| RSCANn(CFD)GTSTCTR register | RTME, ICBCTME |

**Note:** Bits and flags in parentheses exist only in registers in CAN FD mode.

## 21.8 Reception Function

There are two reception types.

- Reception by receive buffers:
  Zero to 96 receive buffers can be shared by all channels. Since messages stored in receive buffers are overwritten at each reception, the latest receive data can always be read.
- Reception by receive FIFO buffers and transmit/receive FIFO buffers (receive mode):
  Eight receive FIFO buffers can be shared by all channels and three dedicated transmit/receive FIFO buffers are provided for each channel. Messages of up to the number of buffer stages specified with the RFDC[2:0] and CFDC[2:0] bits can be stored in FIFO buffers and can be read sequentially from the oldest.

### 21.8.1 Data Processing Using the Receive Rule Table

Data processing using the receive rule table allows dispatching of selected messages to the specified buffer. Data processing includes acceptance filter processing, DLC filter processing, routing processing, label addition processing, and mirror function processing.

Up to 128 receive rules can be registered per channel and up to (64 × number of channels) total receive rules can be registered in the entire module. (Up to 384 receive rules can be registered in this module that has six channels.) Set receive rules for each channel. Receive rules cannot be shared with other channels. If receive rules are not set, no messages can be received. **Figure 21.6** illustrates how receive rules are registered.

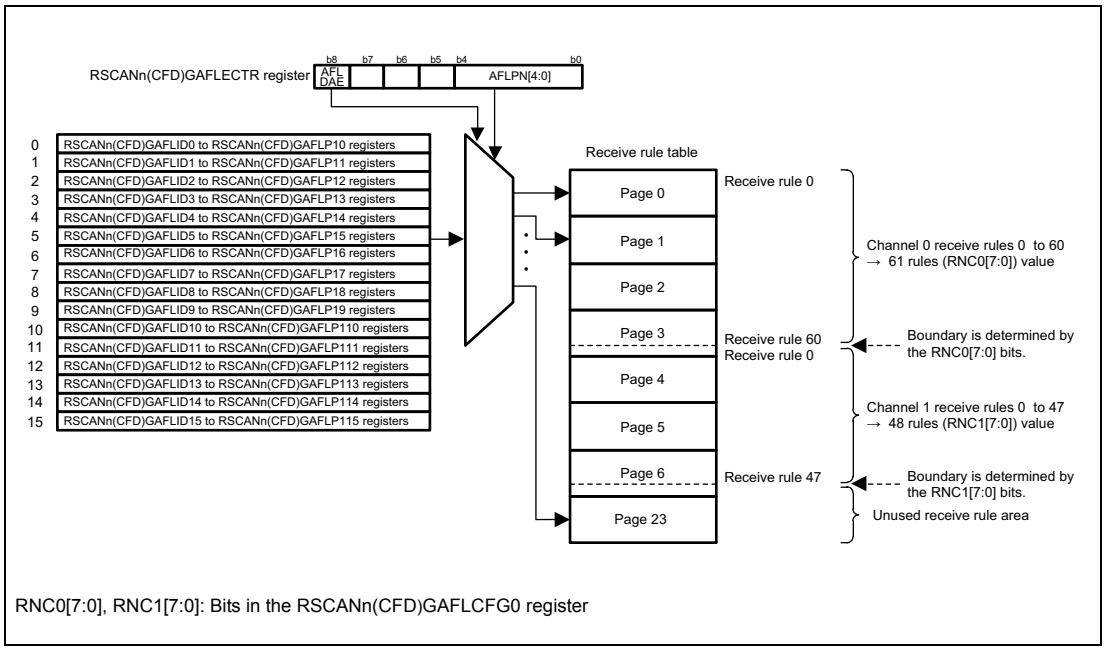



**Figure 21.6 Entry of Receive Rules (for Setting Channel 0 and 1)**

**CAUTION**

**Receive rules for each channel must be set in contiguous blocks.**
**Channel 1 rules and channel 0 rules must be set separately.**

Each receive rule consists of 16 bytes in the RSCANn(CFD)GAFLIDj, RSCANn(CFD)GAFLMj, RSCANn(CFD)GAFLP0j, and RSCANn(CFD)GAFLP1j registers (j = 0 to 15). The RSCANn(CFD)GAFLIDj register is used to set GAFLID, GAFLIDE bit, GAFLRTR bit, and the mirror function, the RSCANn(CFD)GAFLMj register is used to set mask, the RSCANn(CFD)GAFLP0j register is used to set label information to be added, DLC value, and storage receive buffer, and the RSCANn(CFD)GAFLP1j register is used to set storage FIFO buffer. Up to 16 receive rules can be set per page.

### 21.8.1.1 Acceptance Filter Processing

In the acceptance filter processing, the ID data, IDE bit, and RTR bit in a received message are compared with the ID data, IDE bit, and RTR bit set in the receive rule of the corresponding channel. When all these bits match, the message passes through the acceptance filter processing. The ID data, IDE bit, and RTR bit in the received message which correspond to the bits set to 0 (bits are not compared) in the RSCANn(CFD)GAFLMj register are not compared and are regarded as matched.

Check begins with the receive rule of the minimum number for the corresponding channel. When all the bits to be compared in a received message match the bits set in the receive rule or when all the receive rules are compared without any match, filter processing stops. If there is no matching receive rule, the received message is not stored in the receive buffer or FIFO buffer.

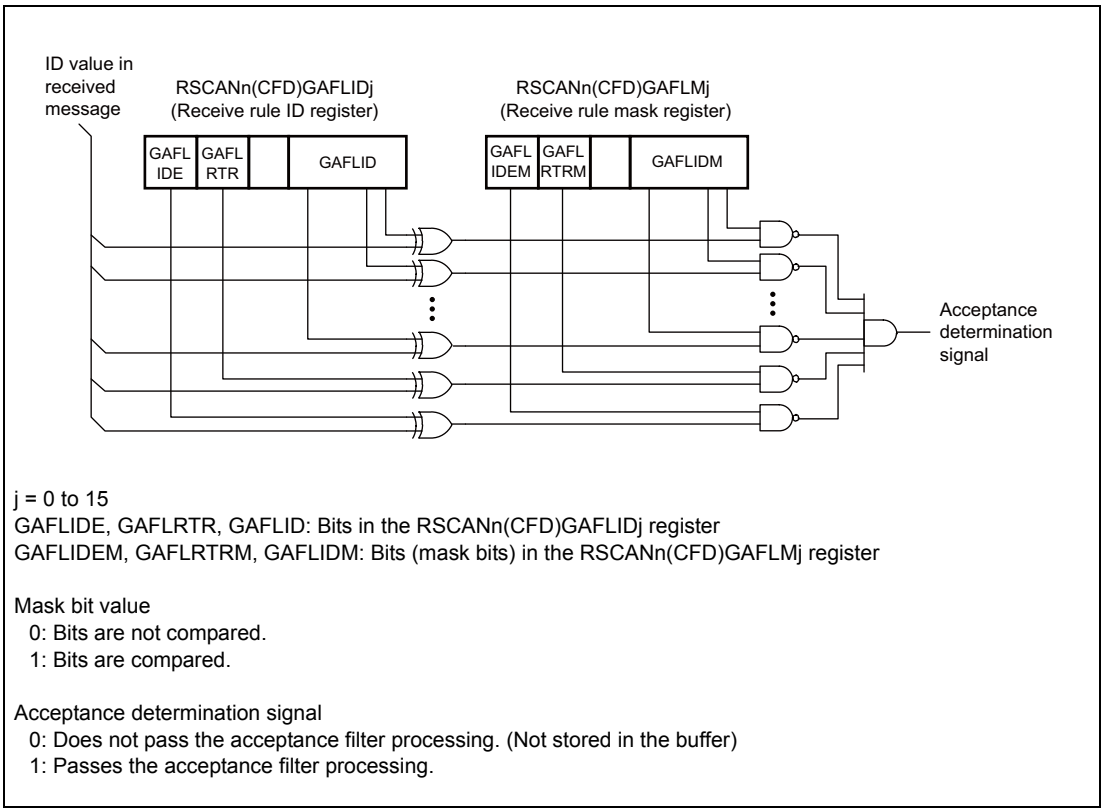



**Figure 21.7 Acceptance Filter Function**

#### 21.8.1.2 DLC Filter Processing

When the DCE bit in the RSCANn(CFD)GCFG register is set to 1 (DLC check is enabled), DLC filter processing is added to messages that passed through the acceptance filter processing. When the DLC value in a message is equal to or larger than the DLC value set in the receive rule, the message passes through the DLC filter processing.

When a message has passed through the DLC filter processing with the DRE bit in the RSCANn(CFD)GCFG register set to 0 (DLC replacement is disabled), the DLC value in the received message is stored in the buffer. In this case, all the data bytes in the received message are stored in the buffer.

When a message has passed through the DLC filter processing with the DRE bit in the RSCANn(CFD)GCFG register set to 1 (DLC replacement is enabled), the DLC value in the receive rule is stored in the buffer instead of the DLC value in the received message. In this case, a value of $00_H$ is stored in each data byte beyond the number of bytes which is indicated by the DLC value in the receive rule.

When the DLC value in the received message is smaller than that in the receive rule, the message does not pass through the DLC filter processing. In this case, the message is not stored in the receive buffer or the FIFO buffer and the DEF flag in the RSCANn(CFD)GERFL register is set to 1 (a DLC error is present).

#### 21.8.1.3 Routing Processing

Messages that passed through the acceptance filter processing and the DLC filter processing are stored in receive buffers, receive FIFO buffers, or transmit/receive FIFO buffers (set to receive mode or gateway mode). Message storage destination is set by the GAFLRMV and GAFLRMDP[6:0] bits in the RSCANn(CFD)GAFLP0j register (j = 0 to 15) and by the RSCANn(CFD)GAFLP1j register. Messages that passed through the acceptance filter processing and the DLC filter processing can be stored in up to eight buffers.

In CAN FD mode, if the payload length of the received message exceeds the payload storage size of the storage buffer, the CMPOF flag in the RSCFDnCFDGERFL register is set to 1 (payload overflow) and the processing is handled according to the CMPOC bit in the RSCFDnCFDGCFG register. When the CMPOC bit is 0, the received message which exceeds the payload storage size is not stored in the buffer. When the CMPOC bit is 1, the received message is stored in the buffer with payloads exceeding the storage size being discarded.

#### 21.8.1.4 Label Addition Processing

It is possible to add 12-bit label information to messages that passed through the filter processing and store them in buffers. This label information is set in the GAFLPTR[11:0] bits in the RSCANn(CFD)GAFLP0j register.

#### 21.8.1.5 Mirror Function Processing

The mirror function allows the CAN node to receive its own transmitted messages. The mirror function is made available by setting the MME bit in the RSCANn(CFD)GCFG register to 1 (mirror function is enabled).

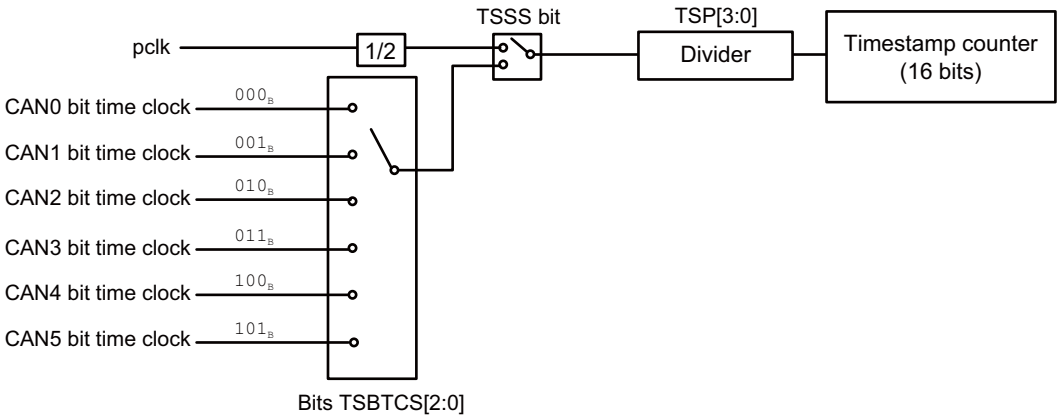
When the mirror function is in use, receive rules for which the GAFLLB bit in the RSCANn(CFD)GAFLIDj register is set to 0 are used for data processing when receiving messages transmitted from other CAN nodes. When the CAN node is receiving its own transmitted messages, receive rules for which the GAFLLB bit is set to 1 are used for data processing.

### 21.8.1.6 Timestamp

The timestamp counter is a 16-bit free-running counter used for recording message receive time and transmission time. The timestamp counter value is fetched at the start-of-frame (SOF) timing of a message and is then stored in a receive buffer or a FIFO buffer together with the message ID and data during data reception. The clock source of the timestamp counter is selected by the TSBTCS[2:0] and TSSS bits in the RSCANn(CFD)GCFG register. In classical CAN mode, either pclk/2 or the CANm bit time clock (m = 0 to 5). In CAN FD mode, the clock source is selectable from pclk/2 or nominal CANm bit time clock. However, do not select the nominal CANm bit time clock of channels that handle CAN FD frames. The timestamp counter count source is obtained by dividing the selected clock source by the TSP[3:0] value in the RSCANn(CFD)GCFG register.

When the CANm bit time clock or nominal CANm bit time clock is used as a clock source, the timestamp counter stops when the corresponding channel transitions to channel reset mode or channel halt mode. When the pclk/2 is used as a clock source, the timestamp function is not affected by channel mode.

The timestamp counter value is reset to $0000_H$ by setting the TSRST bit in the RSCANn(CFD)GCTR register to 1.




TSBTCS[2:0], TSSS, TSP[3:0]: Bits in the RSCANn(CFD)GCFG register

**Note:** When specifying pclk/2 as the timestamp counter count source, set bits TSBTCS[2:0] to $000_B$.

**Figure 21.8 Timestamp Function Block Diagram**

## 21.9 Transmission Functions

There are three types of transmission. In classical CAN mode, transmittable payload length is 8 bytes in every transmission type. In CAN FD mode, transmittable payload length varies with transmission types.

- Transmission using transmit buffers:
  Each channel has 16 buffers. Transmittable payload length in CAN FD mode is 20 bytes. However, when transmit buffer merge mode is used, four buffers out of 16 buffers are allocated as a payload-only storage area and two buffers are able to transmit payloads with a length of more than 20 bytes.
- Transmission using transmit/receive FIFO buffers (transmit mode):
  Each channel has three FIFO buffers. Up to 128 messages can be contained in a single FIFO buffer. Transmittable payload length in CAN FD mode is 64 bytes. Each FIFO buffer is used with a link to a transmit buffer. Only the message to be transmitted next in a FIFO buffer becomes the target of transmit priority determination. Messages are transmitted sequentially on a first-in, first-out basis.
- Transmission using transmit queues:
  Up to 16 transmit buffers per channel can be allocated to the transmit queues. Transmittable payload length in CAN FD mode is 20 bytes. Transmit buffer ((16 × m) + 15) is used as an access window of a corresponding channel. Transmit buffers are allocated to transmit queues in descending order of buffer number. All messages in transmit queues, which are targets of priority determination, are transmitted in the order of ID number.

**Figure 21.9** shows the allocation of transmit queues and transmit/receive FIFO buffer links.

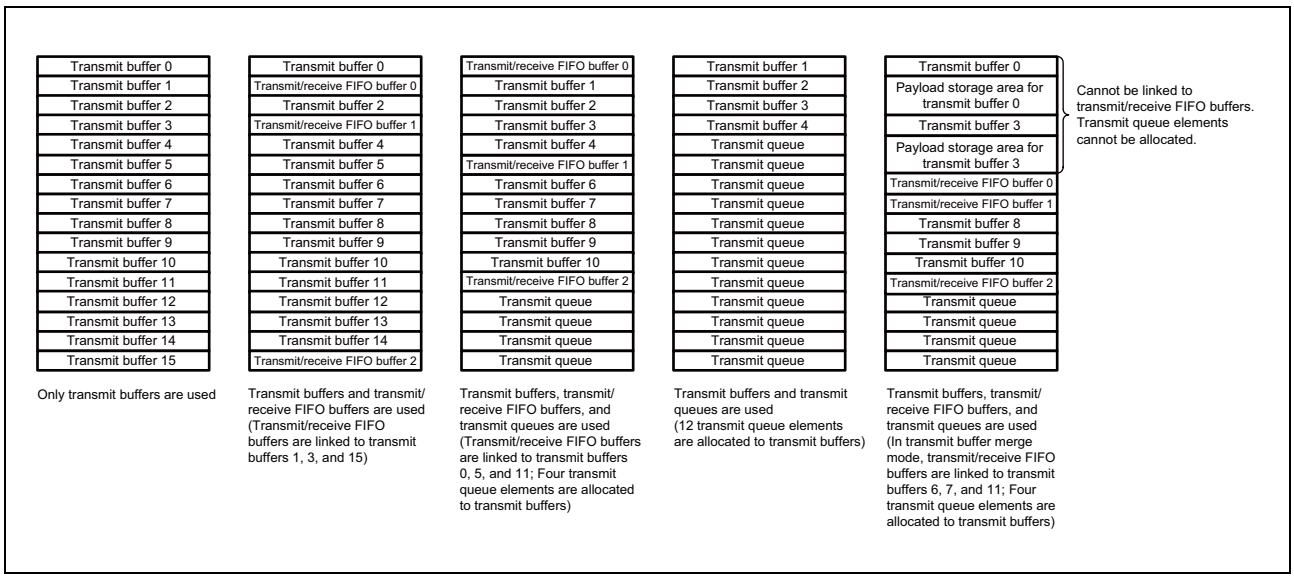



**Figure 21.9 Allocation of Transmit Queues and Transmit/Receive FIFO Buffer Links**

### 21.9.1 Transmit Priority Determination

If transmit requests are issued from multiple buffers or from the queue on the same channel, transmit priority is determined using one of the following methods.

The priority is determined by using one of the following methods.

- ID priority (TPRI bit = 0)
- Transmit buffer number priority (TPRI bit = 1)

All CAN channels use the setting of the TPRI bit in the RSCANn(CFD)GCFG register.

When the TPRI bit is set to 0, messages are transmitted according to the priority of stored message IDs. ID priority conforms to the CAN bus arbitration specification defined in the CAN specifications. All IDs of pending transmit messages are targets of priority determination, regardless of whether they are stored in transmit buffers, transmit/receive FIFO buffers (set to transmit mode or gateway mode), or the transmit queue. If even a single transmit queue is used, select ID priority. When transmit/receive FIFO buffers are used, the oldest message in a FIFO buffer becomes the target of priority determination. When a message is being transmitted from a transmit/receive FIFO buffer, the next message in the FIFO buffer becomes the target of priority determination. When a transmit queue is used, all messages in the transmit queue are targets of priority determination. If the same ID is set for two or more buffers, the buffer with the smaller buffer number takes precedence.

When the TPRI bit is set to 1, the message in the transmit buffer with the minimum buffer number among all buffers with a transmit request is transmitted first. When transmit/receive FIFO buffers are linked to transmit buffers, transmit priority is determined according to linked transmit buffer numbers.

When messages are retransmitted due to an arbitration-lost or an error, transmit priority determination is made again regardless of the TPRI bit. When a 2-bit ECC error is detected in the priority determination processing, no message is transmitted (only when the EEFE bit in the RSCANnGCFG register is 1 in classical CAN mode).

### 21.9.2 Transmission Using Transmit Buffers

Setting the transmit request bit (TMTR bit in the RSCANn(CFD)TMCp register) in a transmit buffer to 1 (transmission is requested) allows transmission of data frames or remote frames.

The transmit result is shown by the TMTRF[1:0] flag in the corresponding RSCANn(CFD)TMSTSp register (p = 0 to 95). When transmit completes successfully, the TMTRF[1:0] flag is set to $10_B$ (transmission has been completed (without transmit abort request)) or $11_B$ (transmission has been completed (with transmit abort request)).

#### 21.9.2.1 Transmit Abort Function

With respect to transmit buffers for which the TMTRM bit in the RSCANn(CFD)TMSTSp register is set to 1 (a transmit request is present), when the TMTAR bit in the RSCANn(CFD)TMCp register is set to 1 (transmit abort is requested), the transmit request is canceled. When transmit abort is completed, the TMTRF[1:0] flag in the RSCANn(CFD)TMSTSp register is set to $01_B$ (transmit abort has been completed) and the transmit request is canceled (clearing the TMTRM bit to 0).

A message that is being transmitted or a message to be transmitted next according to the transmit priority determination cannot be aborted. However, when an arbitration-lost or an error occurs during transmission of a message for which the TMTAR bit is set to 1, retransmission is not performed.

#### 21.9.2.2 One-Shot Transmission Function (Retransmission Disabling Function)

When the TMOM bit in the RSCANn(CFD)TMCp register is set to 1 (one-shot transmission is enabled), transmission is performed only once. Even if an arbitration-lost or an error occurs, retransmission is not performed.

The one-shot transmit result is shown by the TMTRF[1:0] flag in the corresponding RSCANn(CFD)TMSTSp register. When one-shot transmission completes successfully, the TMTRF[1:0] flag is set to $10_B$ or $11_B$. When an arbitration-lost or an error occurs, the TMTRF[1:0] flag is set to $01_B$ (transmit abort has been completed).

#### 21.9.2.3 Transmit Buffer Merge Mode (Only in CAN FD Mode)

Transmit buffers can transmit messages with a payload length of 20 bytes, but can transmit messages with a payload length of up to 64 bytes by merging three transmit buffers in transmit buffer merge mode.

Setting the TMME bit to 1 in the RSCFDnCFDCmFDCFG register enables transmit buffer merge mode. In this mode, six buffers per channel become a merge area and two sets of transmit buffers (16 × m) + 0 to (16 × m) + 2 and transmit buffers (16 × m) + 3 to (16 × m) + 5 are merged. A transmission request is made by the first transmit buffer, and subsequent two buffers are used as a payload storage area. Do not set the transmission request bit (TMTR bit in the RSCFDnCFDTMCp register) and the transmission abort request bit (TMTAR bit in the RSCFDnCFDTMCp register) to 1 for transmit buffers except for the first buffer.

While transmit buffer merge mode is enabled, do not link the transmit/receive FIFO buffer to six merged buffers or allocate it to the transmit queue.

### 21.9.3 Transmission Using FIFO Buffers

Multiple messages can be stored in a single transmit/receive FIFO buffers, up to the number specified by the FIFO buffer depth, which is set by the CFDC[2:0] bits in the RSCANn(CFD)CFCCk register (k = 0 to 17). Messages are transmitted sequentially on a first-in, first-out basis.

Each transmit/receive FIFO buffer is linked to a transmit buffer selected by the CFTML[3:0] bits in the RSCANn(CFD)CFCCk register. When the CFE bit in the RSCANn(CFD)CFCCk register is set to 1 (transmit/receive FIFO buffers are used), transmit/receive FIFO buffers become targets of transmit priority determination. Priority of only the next transmit message is determined in the FIFO buffer.

When the CFE bit is set to 0 (no transmit/receive FIFO buffer is used), the CFEMP flag is set to 1 (the transmit/receive FIFO buffer contains no message (buffer empty)) at the timing below.

- The transmit/receive FIFO buffer becomes empty immediately if the message in it is not being transmitted or is not to be transmitted next.
- The transmit/receive FIFO buffer becomes empty after transmission completion, CAN bus error detection, or arbitration-lost in the case that a message in it is being transmitted or to be transmitted next.

When the CFE bit is cleared to 0, all messages in transmit/receive FIFO buffers are lost and messages cannot be stored in FIFO buffers. Confirm that the CFEMP flag is set to 1 before setting the CFE bit to 1 again.

#### 21.9.3.1 Interval Transmission Function

A message transmission interval time can be set to space the transmission of messages from the same FIFO buffer when using a transmit/receive FIFO buffer set to transmit mode or gateway mode.

Immediately after the first message has been transmitted successfully from the FIFO buffer with the CFE bit in the RSCANn(CFD)CFCCk register set to 1, the interval timer starts counting (after EOF7 of the CAN protocol). After that, when the interval time has passed, the next message is transmitted. The interval timer stops in channel reset mode or by clearing the CFE bit to 0.

The interval time is set by the CFITT[7:0] bits in the RSCANn(CFD)CFCCk register. When the interval timer is not used, set the CFITT[7:0] bits to $00_H$.

Select an interval timer count source using the CFITR and CFITSS bits in the RSCANn(CFD)CFCCk register. When the interval timer is not used, set the CFITT[7:0] bits to $00_H$. Select an interval timer count source using the CFITR bit in the RSCANn(CFD)CFCCk register and the CFITSS bit in the RSCANnCFCCk register. When the CFITR and CFITSS bits are set to $00_B$, the count source is obtained by dividing pclk/2 by the value of the ITRCP[15:0] bits. When the CFITR and CFITSS bits are set to $10_B$, the count source is obtained by dividing pclk/2 by (the value of the ITRCP[15:0] bits × 10). When the CFITR and CFITSS bits are set to $x1_B$, the CANm bit time clock becomes a count source in classical CAN mode and the nominal CANm bit time clock becomes a count source in CAN FD mode.

The interval time is calculated by the following equations where M is the value of ITRCP[15:0] and N is the value of CFITT[7:0].

- When CFITR and CFITSS = $00_B$:

$$\frac{1}{\text{pclk frequency}} \times 2 \times M \times N$$

- When CFITR and CFITSS = $10_B$:

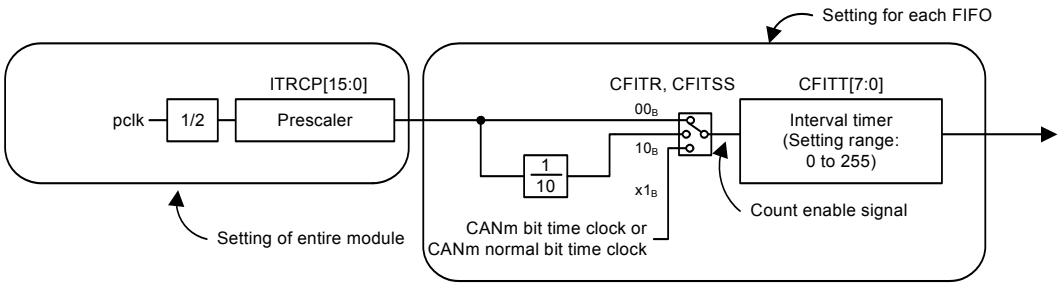
$$\frac{1}{\text{pclk frequency}} \times 2 \times M \times 10 \times N$$

- When CFITR and CFITSS = $x1_B$:

$$\text{Classical CAN mode: } \frac{1}{\text{CANm bit time clock frequency}} \times N$$

$$\text{CAN FD mode: } \frac{1}{\text{Nominal CANm bit time clock frequency}} \times N$$

**Figure 21.10** shows the interval timer block diagram.

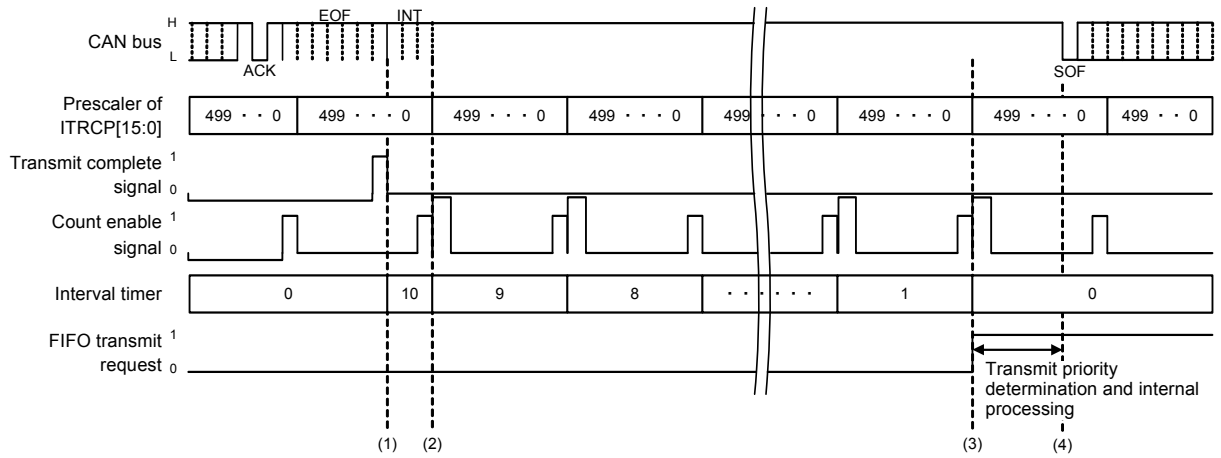



ITRCP[15:0]: Bits in the RSCANn(CFD)GCFG register
CFITR, CFITSS, CFITT[7:0] : Bits in the RSCANn(CFD)CFCCk register

**Note:** m = 0 to 5, k = 0 to 17

**Figure 21.10 Interval Timer Block Diagram**

**Figure 21.11** shows the interval timer timing chart.




$$\text{Interval time (logical value)} = \frac{2}{\text{pclk frequency}} \times (\text{ITRCP[15:0] setting value}) \times \text{CFITT[7:0] setting value}$$

- ITRCP[15:0]: Bits in the RSCANn(CFD)GCFG register (The set value is 500 in this figure.)
- CFITT[7:0]: Bits in the RSCANn(CFD)CFCCk register (The set value is 10 in this figure.)

**Figure 21.11 Interval Timer Timing Chart**

(1) The interval timer starts counting upon completion of transmission. Since the prescaler is not initialized at the time of transmission completion, the first interval time contains an error of up to one count of the interval timer.

(2) The interval timer is decremented by the next count enable signal.

(3) When the interval timer has decreased to 0, the transmit/receive FIFO buffer issues a transmit request.

(4) The transmit/receive FIFO buffer is determined for the next transmission by the priority determination, it starts transmitting data. Transmission starts usually with a delay of three CANm bit time clock cycles or less from the issue of transmit request. If multiple internal processes (such as receive filter processing, message routing, and transmit priority determination) take place in all channels, a delay of up to 1164 cycles of the pclk may be generated.

### 21.9.4 Transmission Using Transmit Queues

Three to sixteen buffers (up to 10 buffers in transmit buffer merge mode) are allocated to a transmit queue for each channel, and transmit buffer ((16 × m) + 15) is used as an access window of a corresponding channel.

All messages in a transmit queue are targets of transmit priority determination and are transmitted in the ID priority order regardless of storage sequence. If two messages having the same ID are stored in a transmit queue, these messages are not always transmitted in the order of their storage in the transmit queue.

Setting the TXQE bit in the RSCANn(CFD)TXQCCm register to 0 disables transmit queues. When the TXQE bit is set to 0, the TXQEMP flag in the RSCANn(CFD)TXQSTSm register is set to 1 (the transmit queue contains no messages (transmit queue empty)) at the timing below.

- The transmit queue becomes empty immediately when no message in it is being transmitted or will be transmitted next.
- The transmit queue becomes empty after transmission completion, CAN bus error detection, or arbitration-lost when a message in it is being transmitted or will be transmitted next.

When the TXQE bit is cleared to 0, all messages in transmit queues are lost and messages cannot be stored in transmit queues. Confirm that the TXQEMP flag is set to 1 before setting the TXQE bit to 1 again.

### 21.9.5 Transmit Data Padding (Only in CAN FD Mode)

When the payload length indicated by the DLC setting value in a transmit message exceeds the payload storage area size of a buffer to be used for transmission, excessive payloads are padded by $CC_H$.

This processing is performed in the following cases when the transmit buffer merge mode is disabled (TMME bit in the RSCFDnCFDCmFDCFG register is 0).

- Transmit/receive FIFO set to transmission or gateway mode:
  When the payload length of the transmit DLC exceeds the transmit/receive FIFO payload storage area size set by the CFPLS[2:0] bits in the RSCFDnCFDCFCCk register
- Transmit buffer (including transmit queue):
  When the payload length of the transmit DLC exceeds 20 bytes

When the transmit buffer merge mode is enabled, no transmit data is padded in any transmission using a transmit buffer, transmit/receive FIFO buffer, or transmit queue. At this time, do not set a payload length more than the payload storage size of the buffer for transmitting as the DLC value in the transmit message.

### 21.9.6 Transmit History Function

Information about transmission-completed messages can be stored in the transmit history buffer. Each channel has a single transmit history buffer that can contain 16 sets of transmit history data.

A message transmit source buffer type can be selected by the THLDTE bit in the RSCANn(CFD)THLCCm register. The THLEN bit in the RSCANn(CFD)CFIDk register (k = 0 to 17) determines whether transmit history data is stored for each message.

In classical CAN mode, the TMTSCE bit in the RSCANnGCFG register can be used to set whether to include a timestamp value in the transmit history data. In CAN FD mode, a timestamp value is always included.

The following information on a transmitted message will be stored in the transmit history buffer after the successful completion of transmission.

Storage of the transmit history data after the successful completion of transmission may take up to 152 cycles of pclk in classical CAN mode or 420 cycles of pclk in CAN FD mode.

- Buffer type — $001_B$: Transmit buffer<br>$010_B$: Transmit/receive FIFO buffer<br>$100_B$: Transmit queue
- Buffer number — Number of source transmit buffer, transmit queue, or transmit/receive FIFO buffer. This number depends on buffer types. See **Table 21.184**.
- Label data — Label information of the transmit message
- Timestamp — Timestamp value of the transmit message<br>(When the TMTSCE bit is 1 in classical CAN mode)

**Table 21.184 Transmit History Data Buffer Numbers**

| Buffer No. / Buffer type | $001_B$ | $010_B$ | $100_B$ |
|---|---|---|---|
| $0000_B$ | Transmit buffer 16 × m + 0 | Buffer numbers of the transmit buffer linked to the transmit/receive FIFO buffer by the CFTML[3:0] bits in the RSCANn(CFD)CFCCk register<br>(k = 0 to 17) | Buffer numbers of the transmit buffer allocated to the transmit queue that performed transmission |
| $0001_B$ | Transmit buffer 16 × m + 1 | | |
| $0010_B$ | Transmit buffer 16 × m + 2 | | |
| $0011_B$ | Transmit buffer 16 × m + 3 | | |
| $0100_B$ | Transmit buffer 16 × m + 4 | | |
| $0101_B$ | Transmit buffer 16 × m + 5 | | |
| $0110_B$ | Transmit buffer 16 × m + 6 | | |
| $0111_B$ | Transmit buffer 16 × m + 7 | | |
| $1000_B$ | Transmit buffer 16 × m + 8 | | |
| $1001_B$ | Transmit buffer 16 × m + 9 | | |
| $1010_B$ | Transmit buffer 16 × m + 10 | | |
| $1011_B$ | Transmit buffer 16 × m + 11 | | |
| $1100_B$ | Transmit buffer 16 × m + 12 | | |
| $1101_B$ | Transmit buffer 16 × m + 13 | | |
| $1110_B$ | Transmit buffer 16 × m + 14 | | |
| $1111_B$ | Transmit buffer 16 × m + 15 | | |

Label data is used to identify each message. Unique label data can be added to each message transmitted from a transmit buffer, transmit queue, or transmit/receive FIFO buffer.

The timestamp value is fetched from the timestamp counter at the SOF (start of frame) timing of the message. For details about the timestamp counter, see **Section 21.8.1.6, Timestamp**.

Transmit history data can be read from the RSCANn(CFD)THLACCm register. If an attempt is made to store new transmit history data while the buffer is full, the buffer overflows and the new data is discarded.

## 21.10 Test Function

The test function is classified into communication tests and global tests.

- Communication tests: Performed for each channel.
  - Standard test mode
  - Listen-only mode
  - Self-test mode 0 (external loopback mode)
  - Self-test mode 1 (internal loopback mode)
  - Restricted operation mode (only in CAN FD mode)
- Global tests: Performed for the entire module
  - RAM test (read/write test)
  - Inter-channel communication test [CRC error test enabled]

### 21.10.1 Standard Test Mode

CRC tests are enabled in standard test mode. The CRC value calculated by the RS-CANFD module based on the transmit message or receive message is stored in the register. This CRC value is stored in the CRCREG[14:0] bits in the RSCANn(CFD)CmERFL register when the message is a classical CAN frame (CRC length = 15 bits) or in the CRCREG[20:0] bits in the RSCFDnCFDCmFDCRC register when the message is a CAN FD frame (CRC length = 17 or 21 bits). Use the inter-channel communication test function for CRC error tests. For details, see **Section 21.10.6.1, CRC Error Test**.

### 21.10.2 Listen-Only Mode

Listen-only mode allows reception of data frames and remote frames. Only recessive bits are transmitted on the CAN bus, and the ACK bit, overload flag, and active error flag are not transmitted.

Listen-only mode is available for detecting the communication speed.

Do not make a transmit request from any buffer or queue in listen-only mode.

**Figure 21.12** shows the connection when listen-only mode is selected.

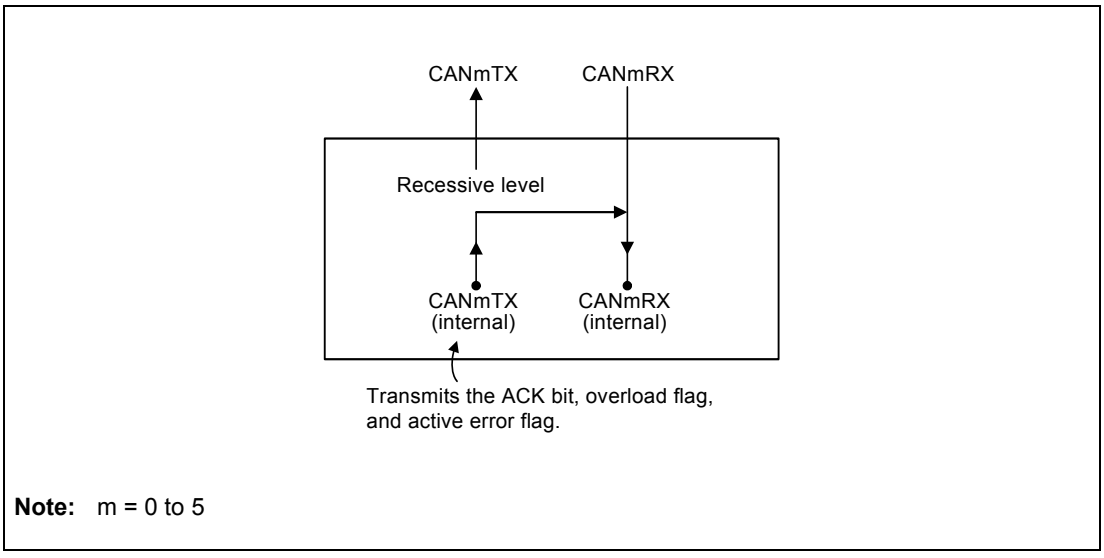



**Figure 21.12 Connection when Listen-Only Mode is Selected**

### 21.10.3 Self-Test Mode (Loopback Mode)

In self-test mode, transmitted messages are compared with the receive rule of the own channel and the messages are stored in a buffer if they have passed through the filter processing. Messages transmitted from other CAN nodes are compared only with the receive rule for which the GAFLLB bit in the RSCANn(CFD)GAFLIDj register (j = 0 to 15) is set to 0 (when a message transmitted from another CAN node is received).

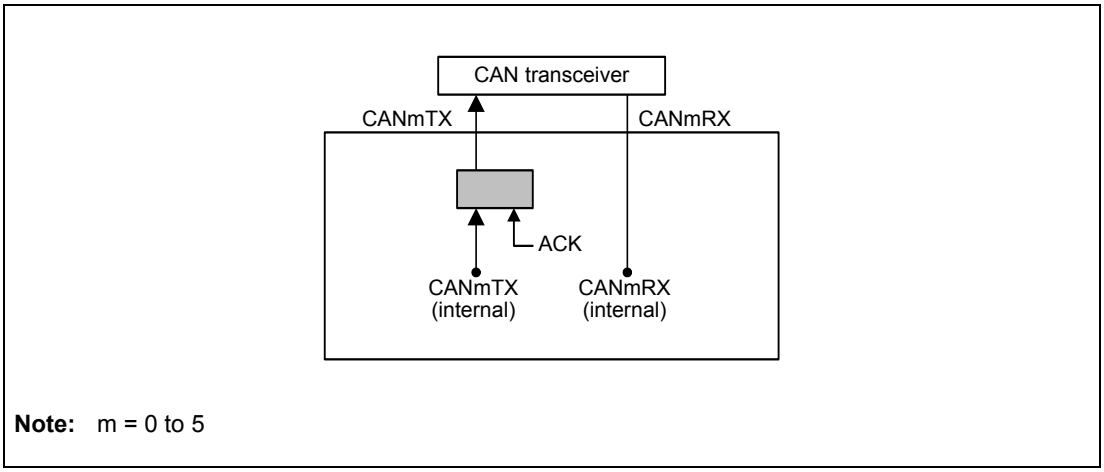
If the mirror function and self-test mode are both enabled, the self-test mode setting takes precedence.

#### 21.10.3.1 Self-Test Mode 0 (External Loopback Mode)

Self-test mode 0 is used to perform a loopback test within a channel including the CAN transceiver.

In self-test mode 0, transmitted messages are handled as messages received through the CAN transceiver and are stored in a buffer. An ACK bit is generated to receive messages transmitted from the own CAN node.

**Figure 21.13** shows the connection when self-test mode 0 is selected.




**Note:** m = 0 to 5

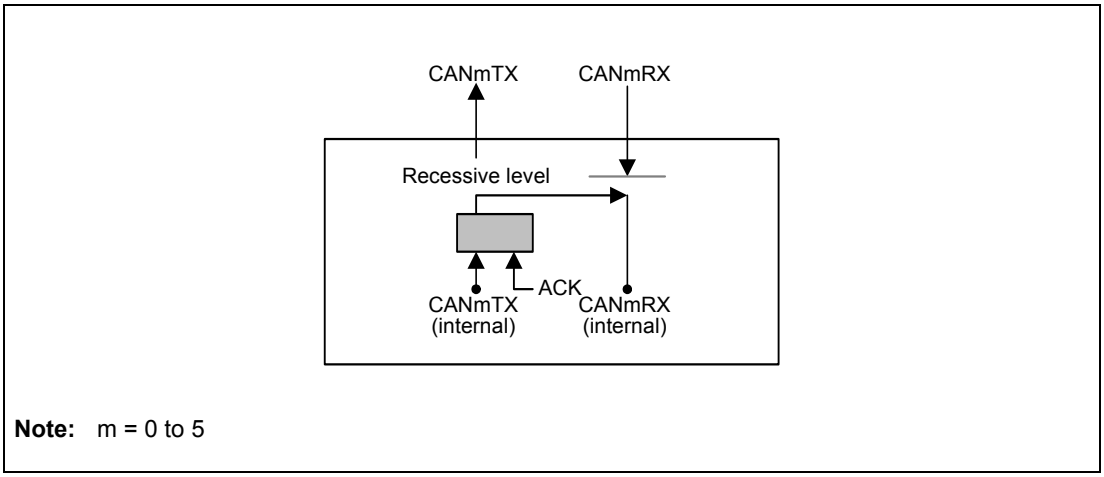
**Figure 21.13 Connection when Self-Test Mode 0 is Selected**

#### 21.10.3.2 Self-Test Mode 1 (Internal Loopback Mode)

In self-test mode 1, transmitted messages are handled as received messages and are stored in a buffer. An ACK bit is generated to receive messages transmitted from the own CAN node.

In self-test mode 1, internal feedback from the internal CANmTX pin (m = 0 to 5) to the internal CANmRX pin is performed. The external CANmRX pin input is isolated. The external CANmTX pin outputs only recessive bits.

**Figure 21.14** shows the connection when self-test mode 1 is selected.




**Note:** m = 0 to 5

**Figure 21.14 Connection when Self-Test Mode 1 is Selected**

### 21.10.4 Restricted Operation Mode (Only in CAN FD Mode)

In restricted operation mode, an ACK bit is generated when a valid data frame and a remote frame have been received, but these frames are not transmitted even if an error frame or an overload frame transmit condition is detected. When a condition is detected, operation is suspended until the state becomes bus idle for resynchronization with the CAN communication. The receive error counter (REC) and the transmit error counter (TEC) do not change due to an error.

A desired transmission request can be made for transmission without restrictions.

### 21.10.5 RAM Test

The RAM test function allows accesses to all CAN RAM addresses.

When the RAM test function is used, the RAM is divided into pages of 256 bytes each. RAM test page is set by the RTMPS[6:0] bits in the RSCANn(CFD)GTSTCFG register. Data in the set page can be read from and written to the RSCANn(CFD)RPGACCr register (r = 0 to 63). The available total RAM size is 15360 bytes ($3C00_H$) in classical CAN mode or 21312 bytes ($5340_H$) in CAN FD mode.

In CAN FD mode, RAM addresses after the first 64 bytes of the last page (the page specified by RTMPS = $53_H$) cannot be accessed.

### 21.10.6 Inter-Channel Communication Test

The inter-channel communication test function allows communication test by internally connecting CAN channels to each other. During this test, channels are isolated from the external CAN bus.

Before starting data transmission/reception in channel communication mode, make transmission/reception settings for each channel.

**Figure 21.15** shows the connection for inter-channel communication test.

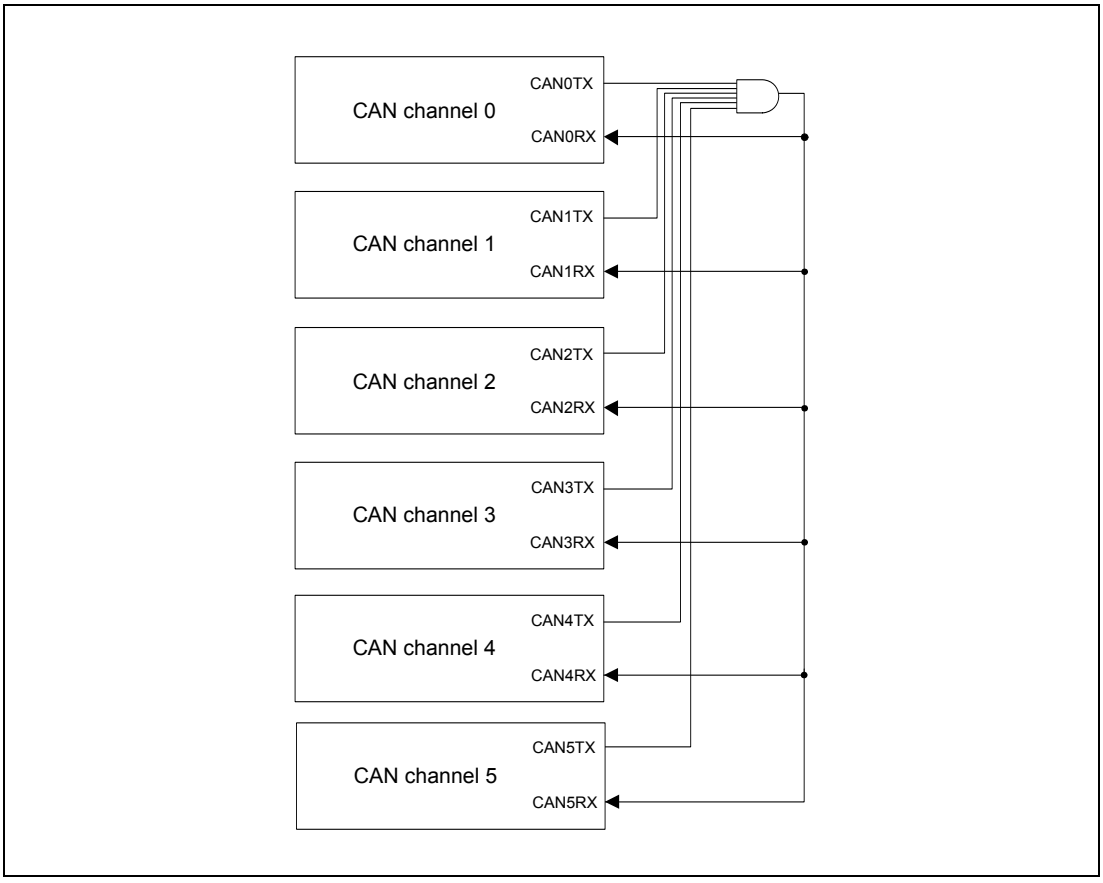



**Figure 21.15 Connection for Inter-Channel Communication Test**

### 21.10.6.1 CRC Error Test

A CRC error test is enabled during an inter-channel communication test. The following shows an example of channel 0 CRC error test procedure during a communication test between channel 0 and channel 1.

#### Preconditions

- Inter-channel communication test is enabled.
- Channel 0 and channel 1 are in standard test mode.

#### Procedure

1. Make a setting to send a message from the transmit buffer p of channel 1.
2. Set the CRCT bit in the RSCANn(CFD)C0CTR register to 1 (to enable inversion of the first bit in the received ID field).
3. Set the TMTR bit in the RSCANn(CFD)TMCp register to 1 (to issue a transmission request to the transmit buffer p of channel 1).
4. Wait for occurrence of a CAN0 error interrupt due to a channel bus error.
5. Read the CRCREG[14:0] bits in the RSCANn(CFD)CmERFL register or the CRCREG[20:0] bits in the RSCFDnCFDCmFDCRC register of channel 0 and channel 1, and confirm that the CRC values are different on the transmission and the reception side.
6. Confirm that the CERR bit in RSCANm(CFD)C0ERFL register is 1 (CRC error detected).

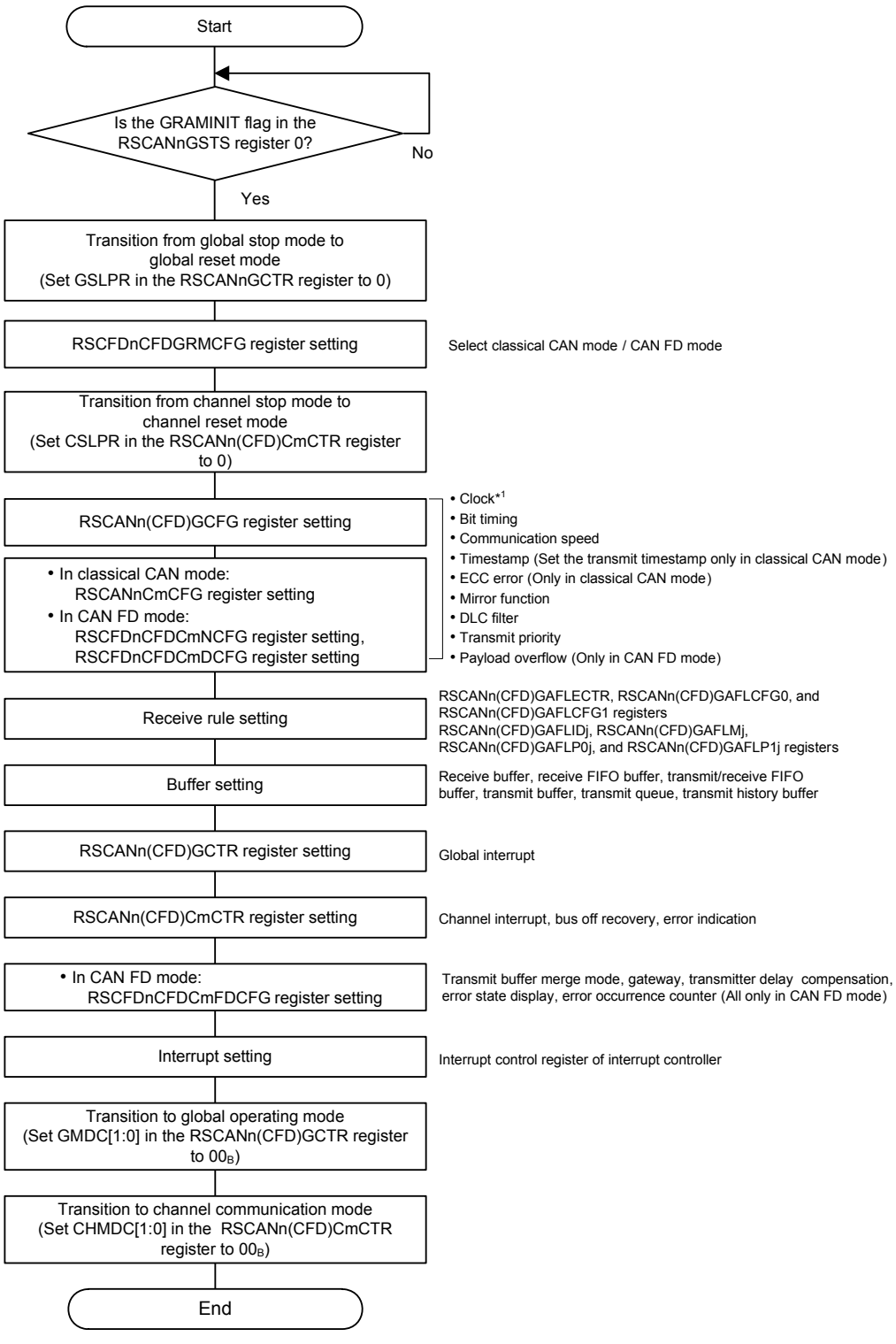
The CRC error test function generates an incorrect CRC value by inverting the first bit in the received ID field. Therefore, note that not a CRC error but a stuff error (continuous 6-bit data of the same level) is detected when a message in which ID's upper 5-bit value is $10000_B$ or ID's upper 6-bit value is $011111_B$ is received.

The CRC generation circuit of the RS-CANFD module is contained in the protocol controller of each channel. Another CRC calculation test is not necessary during transmission because the same circuit is used for both transmission and reception.

# 21.11 RS-CANFD Setting Procedure

## 21.11.1 Initial Settings

The RS-CANFD module initializes the CAN RAM after the MCU is reset. The RAM initialization time is 7586 cycles of the pclk. The GRAMINIT flag in the RSCANn(CFD)GSTS register is set to 1 (CAN RAM initialization is ongoing) during the RAM initialization and is cleared to 0 (CAN RAM initialization is finished) when the initialization is completed. Make CAN settings after the GRAMINIT flag is cleared to 0. **Figure 21.16** shows the CAN setting procedure after the MCU is reset.




Note 1. For the setting of CAN clock frequency, see **Table 21.8, Range of Operating Frequency Depending on the Transfer Rate and the Number of Channels in Use in the RH850/F1K**.

**Note:** m = 0 to 5, j = 0 to 15

**Figure 21.16 CAN Setting Procedure after the MCU is Reset**

### 21.11.1.1 Clock Setting

Set the CAN clock (fCAN) as a clock source of the RS-CANFD module. Select the clk_xincan or clkc using the DCS bit in the RSCANn(CFD)GCFG register.

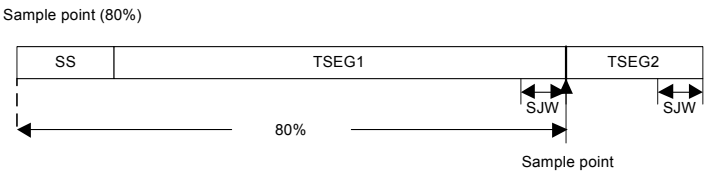
### 21.11.1.2 Bit Timing Setting

In the CAN protocol, one bit of a communication frame consists of three segments SS, TSEG1, and TSEG2, of which two segments TSEG1 and TSEG2 can be set by the corresponding registers for each channel. In classical CAN mode, set these two segments in the RSCANnCmCFG register. Two bit rates (nominal bit rate and data bit rate) are provided for CAN FD mode. Set the nominal bit rate in the RSCFDnCFDCmNCFG register and set the data bit rate in the RSCFDnCFDCmDCFG register. Sample point timing can be determined by setting these two segments. This timing can be adjusted in units of 1 Time Quantum (hereafter referred to as Tq). A single Tq is the cycle of clock obtained by dividing the clock selected by the DCS bit in the RSCANnGCFG register. Set a division ratio by the BRP[9:0] bits in the RSCANnCmCFG register in classical CAN mode (CANmTq clock), and by the NBRP[9:0] bits in the RSCFDnCFDCmNCFG register and the DBRP[7:0] bits in the RSCFDnCFDCmDCFG register in CAN FD mode (CANmTq(N) clock and CANmTq(D) clock).

The NBRP[9:0] value and the DBRP[7:0] value should be equal and the two bit rate values be different according to the respective segment values.

When the TDCE bit is set to 1 (Transmitter delay compensation is enabled) in the RSCFDnCFDCmFDCFG register, set the equal value of 1 or less to the bits NBRP[9:0] and DBRP[7:0]

**Figure 21.17** shows the bit timing chart. **Table 21.185** shows an example of bit timing setting.




| | | |
|---|---|---|
| Classical CAN mode: | SS = 1 Tq fixed<br>Set TSEG1 to the range of 4 Tq to 16 Tq<br>Set TSEG2 to the range of 2 Tq to 8 Tq<br>Set SJW to the range of 1 Tq to 4 Tq<br>Set SS + TSEG1 + TSEG2 to a range of 8 Tq to 25 Tq<br>TSEG1 > TSEG2 ≥ SJW | |
| CAN FD mode: | <Nominal bit rate><br>SS = 1 Tq fixed<br>Set TSEG1 to the range of 4 Tq to 128 Tq<br>Set TSEG2 to the range of 2 Tq to 32 Tq<br>Set SJW to the range of 1 Tq to 32 Tq<br>Set SS + TSEG1 + TSEG2 to a range of 8 Tq to 161 Tq<br>TSEG1 > TSEG2 ≥ SJW | <Data bit rate><br>SS = 1 Tq fixed<br>Set TSEG1 to the range of 2 Tq to 16 Tq<br>Set TSEG2 to the range of 2 Tq to 8 Tq<br>Set SJW to the range of 1 Tq to 8 Tq<br>Set SS + TSEG1 + TSEG2 to a range of 5 Tq to 25 Tq<br>TSEG1 ≥ TSEG2 ≥ SJW |

- SS (synchronization segment):
  The SS is a segment that performs synchronization by monitoring the edge from recessive to dominant bits in the Interframe Space. Interframe Space consists of Intermission, Suspend Transmission, and Bus Idle. All nodes can start transmission during Bus Idle.
- TSEG1 (Time segment 1):
  TSEG1 is a segment that absorbs physical delays on the CAN network. The physical delay on the network is the doubled value of the total of a CAN bus delay, input converter delay, and output driver delay on the CAN bus.
- TSEG2 (Time segment 2):
  TSEG2 is a segment that compensates phase error due to an error in frequency.
- SJW (Resynchronization jump width):
  The SJW is a length to extend or reduce the time segment to compensate for an error in phase due to phase error.

**Figure 21.17 Bit Timing Chart**

**Table 21.185 Example of Bit Timing Setting**

| 1 Bit | Set Value (Tq) | | | | Sample Point (%) Note: See Figure 21.17. |
|---|---|---|---|---|---|
| | SS | TSEG1 | TSEG2 | SJW | |
| 5 Tq*1 | 1 | 2 | 2 | 1 | 60.00 |
| 8 Tq | 1 | 4 | 3 | 1 | 62.50 |
| | 1 | 5 | 2 | 1 | 75.00 |
| 10 Tq | 1 | 6 | 3 | 1 | 70.00 |
| | 1 | 7 | 2 | 1 | 80.00 |
| 16 Tq | 1 | 10 | 5 | 1 | 68.75 |
| | 1 | 11 | 4 | 1 | 75.00 |
| 20 Tq | 1 | 12 | 7 | 1 | 65.00 |
| | 1 | 13 | 6 | 1 | 70.00 |
| 50 Tq*1 | 1 | 39 | 10 | 4 | 80.00 |

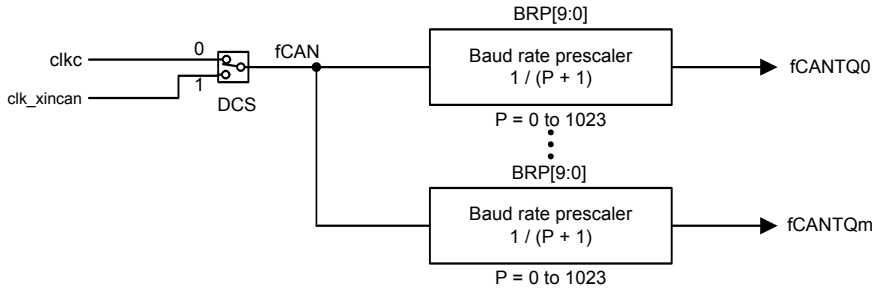
Note 1. Only in CAN FD mode

### 21.11.1.3 Communication Speed Setting

Set the CAN communication speed for each channel using the fCAN, baud rate prescaler division value, and Tq count per bit time. For CAN FD mode, set two types of transmission rate (arbitration phase and data phase) for each channel.

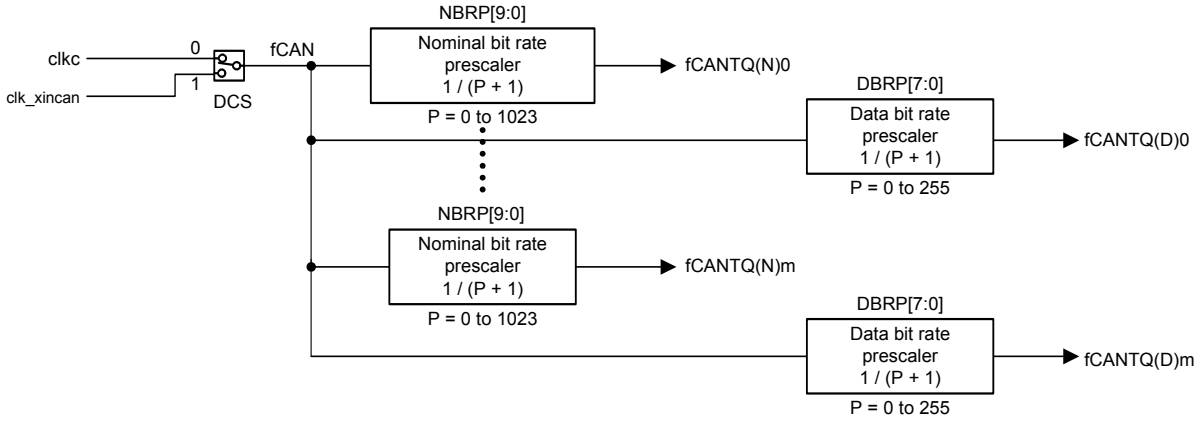
**Figure 21.18** shows the CAN clock control block diagram, and **Table 21.186** and **Table 21.187** show examples of the communication speed setting.

- Classical CAN mode




$$\text{Communication speed} = \frac{\text{fCAN}}{\text{Baud rate prescaler division value} \times (\text{Tq count of 1 bit time})}$$

- CAN FD mode




$$\text{Nominal bit rate} = \frac{\text{fCAN}}{\text{Nominal bit rate prescaler division value} \times (\text{Tq count of 1 nominal bit time})}$$

$$\text{Data bit rate} = \frac{\text{fCAN}}{\text{Data bit rate prescaler division value} \times (\text{Tq count of 1 data bit time})}$$

**Note:** m = 0 to 5
DCS: Bit in the RSCANn(CFD)GCFG register
BRP[9:0]: Bits in the RSCANnCmCFG register
NBRP[9:0]: Bits in the RSCFDnCFDCmNCFG register
DBRP[7:0]: Bits in the RSCFDnCFDCmDCFG register
fCAN: CAN clock
fCANTQm: CANmTq clock
fCANTQ(N)m: CANmTq(N) clock
fCANTQ(D)m: CANmTq(D) clock

**Figure 21.18 CAN Clock Control Block Diagram**

**Table 21.186 Example of Communication Speed Setting (Classical CAN mode)**

| fCAN / Communication Speed | 40 MHz | 32 MHz | 24 MHz | 20 MHz | 16 MHz | 8 MHz |
|---|---|---|---|---|---|---|
| 1 Mbps | 8 Tq (5)<br>20 Tq (2) | 8 Tq (4)<br>16 Tq (2) | 8 Tq (3)<br>12 Tq (2)<br>24 Tq (1) | 10 Tq (2)<br>20 Tq (1) | 8 Tq (2)<br>16 Tq (1) | 8 Tq (1) |
| 500 Kbps | 8 Tq (10)<br>20 Tq (4) | 8 Tq (8)<br>16 Tq (4) | 8 Tq (6)<br>12 Tq (4)<br>24 Tq (2) | 10 Tq (4)<br>20 Tq (2) | 8 Tq (4)<br>16 Tq (2) | 8 Tq (2)<br>16 Tq (1) |
| 250 Kbps | 8 Tq (20)<br>20 Tq (8) | 8 Tq (16)<br>16 Tq (8) | 8 Tq (12)<br>12 Tq (8)<br>24 Tq (4) | 10 Tq (8)<br>20 Tq (4) | 8 Tq (8)<br>16 Tq (4) | 8 Tq (4)<br>16 Tq (2) |
| 125 Kbps | 8 Tq (40)<br>20 Tq (16) | 8 Tq (32)<br>16 Tq (16) | 8 Tq (24)<br>12 Tq (16)<br>24 Tq (8) | 10 Tq (16)<br>20 Tq (8) | 8 Tq (16)<br>16 Tq (8) | 8 Tq (8)<br>16 Tq (4) |

**Table 21.187 Example of Communication Speed Setting (Nominal Bit Rate and Data Bit Rate in CAN FD Mode)**

| fCAN / Communication Speed | 40 MHz | 20 MHz |
|---|---|---|
| Nominal bit rate 1 Mbps<br>Data bit rate 5 Mbps | Nominal bit rate 40 Tq (1)<br>Data bit rate 8 Tq (1) | None |
| Nominal bit rate 500 kbps<br>Data bit rate 2 Mbps | Nominal bit rate 80 Tq (1)<br>Data bit rate 20 Tq (1) | Nominal bit rate 40 Tq (1)<br>Data bit rate 10 Tq (1) |

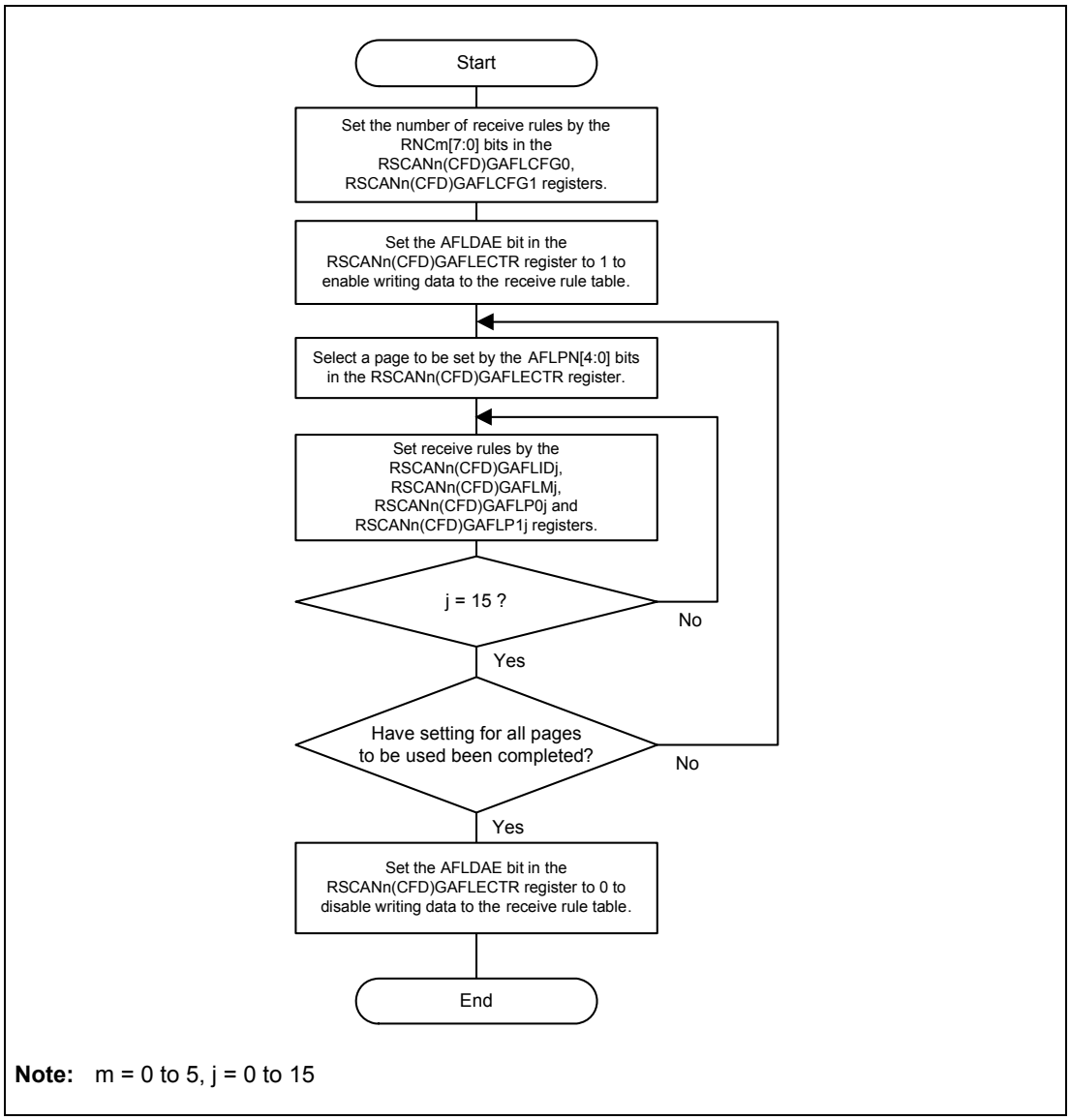
**Note:** Values in ( ) are baud rate prescaler division values.

### 21.11.1.4 Receive Rule Setting

Receive rules can be set using receive rule-related registers.

Up to 16 receive rules can be registered per page. Specify pages 0 to 23 (for 6-channel unit) by the AFLPN[4:0] bits in the RSCANn(CFD)GAFLECTR register. Set receive rule table write enable/ disable using the AFLDAE bit.

**Figure 21.19** shows the receive rule setting procedure.




**Note:** m = 0 to 5, j = 0 to 15

**Figure 21.19 Receive Rule Setting Procedure**

### 21.11.1.5 Buffer Setting

Set the number of buffers to be used (number of messages to be stored) and interrupt sources. Also set the payload storage size for CAN FD mode. For transmit/receive FIFO buffers that are set to transmit mode, set transmit buffers to be linked.

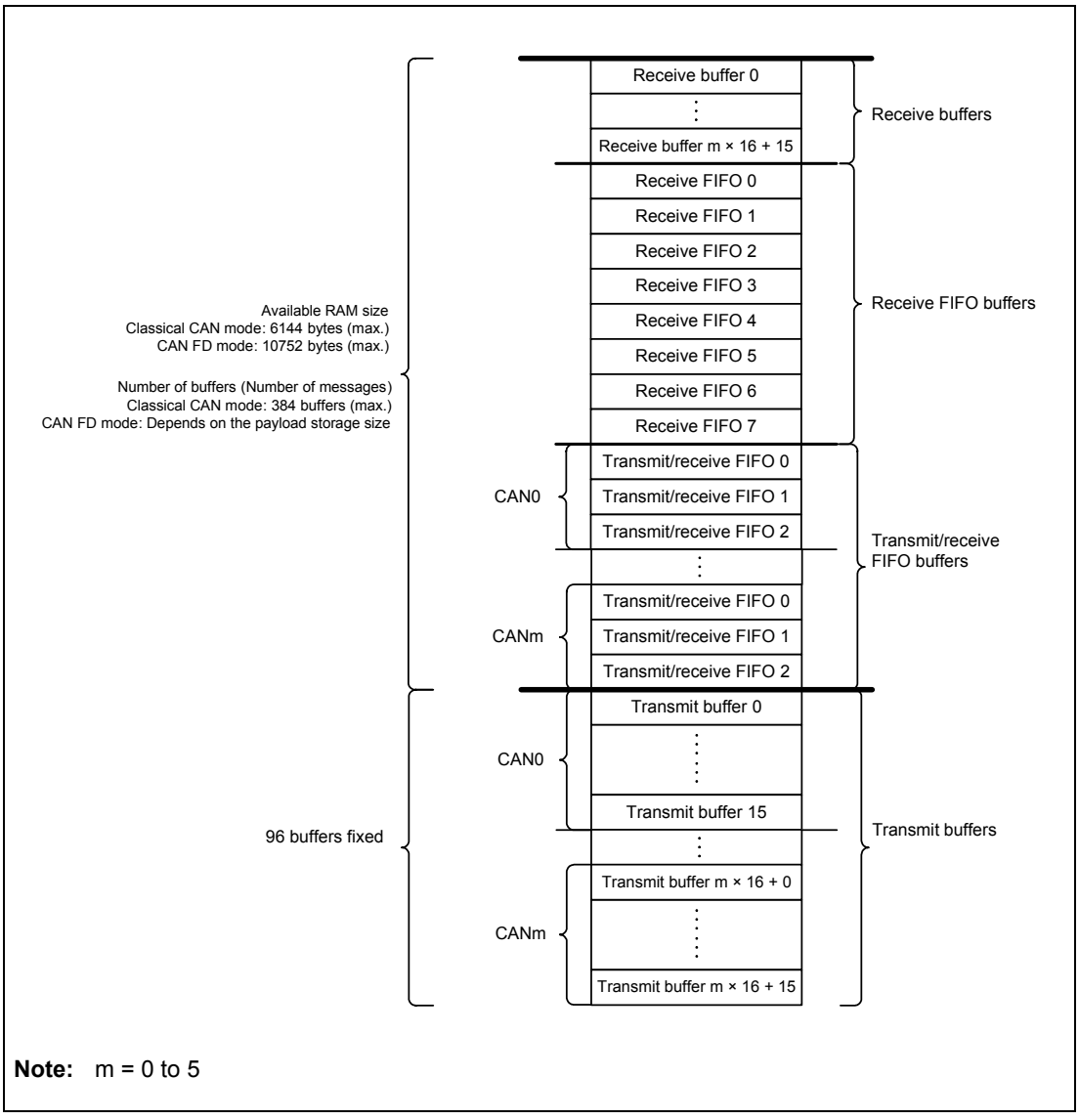
In classical CAN mode, up to 6144 bytes of the RAM can be used in receive buffers and FIFO buffers. Up to 384 buffers are available, and 16 bytes are used per buffer. Configure the buffers so that the following conditions are met.

Number of receive buffers + total number of depth of receive FIFO buffers x + total number of depth of transmit/receive FIFO buffers k ≤ 384 buffers

In CAN FD mode, up to 10752 bytes of the RAM can be used in receive buffers and FIFO buffers. Configure the buffers so that the following conditions are met.

Number of receive buffers × (12 + payload storage size) + total of (number of depth × (12 + payload storage size)) of receive FIFO buffers x + total of (number of depth × (12 + payload storage size)) of transmit/receive FIFO buffers k ≤ 10752 bytes

**Figure 21.20** shows the buffer configuration. **Figure 21.21** shows the buffer setting procedure.

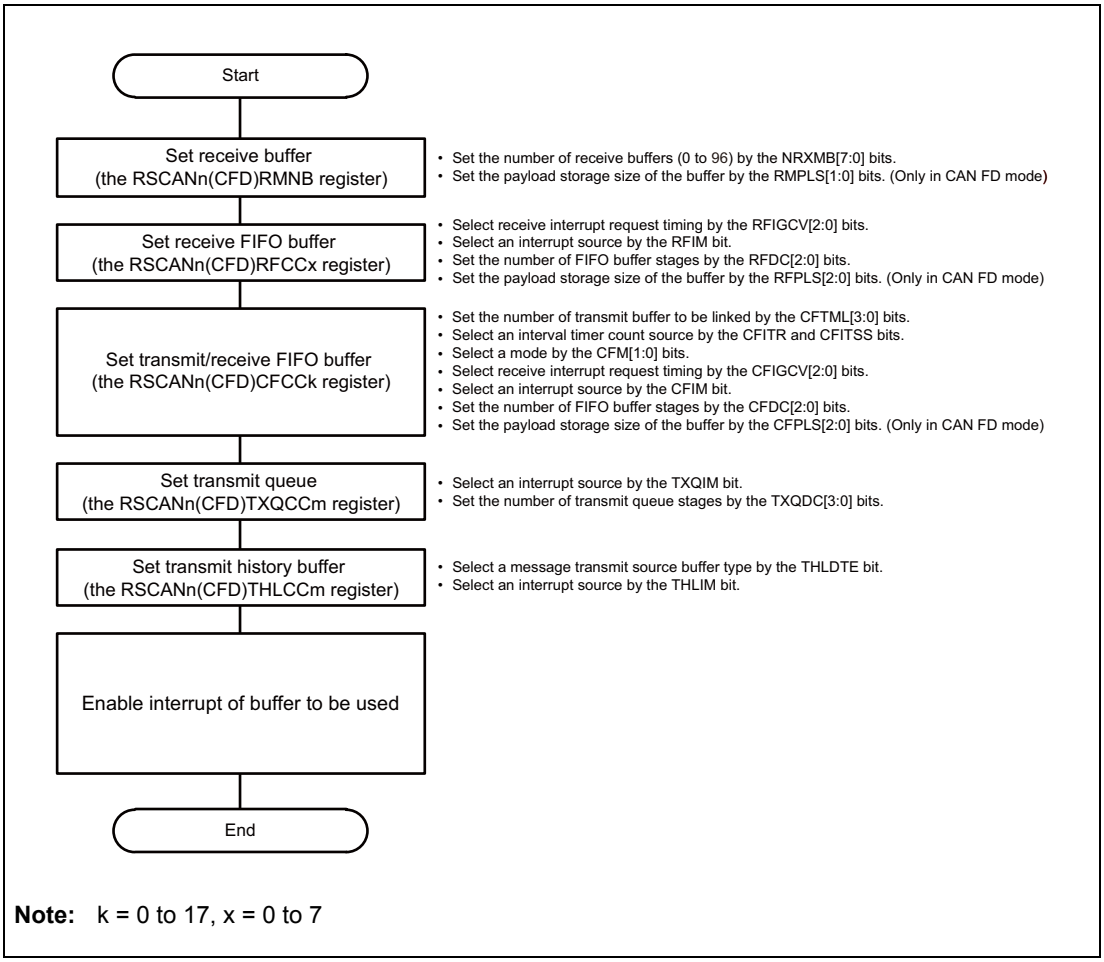



**Note:** m = 0 to 5

**Figure 21.20 Buffer Configuration**

**CAUTION**

**Receive buffers, receive FIFO buffers, transmit/receive FIFO buffers, and transmit buffers are located in succession.**




**Note:** k = 0 to 17, x = 0 to 7

**Figure 21.21 Buffer Setting Procedure**

### 21.11.1.6 Transmitter Delay Compensation (Only in CAN FD Mode)

A high baud rate is used in CAN FD mode. Transmitter delay compensation is provided as a function to accept propagation delay in this case.

To use this function, set the TDCE bit in the RSCFDnCFDCmFDCFG register to 1. Also set the secondary sample point (SSP) timing used in the data phase by the TDCOC bit and TDCO[6:0] bits in the RSCFDnCFDCmFDCFG register.

When the TDCOC bit is 0, the SSP timing equals the total value of the delay measured by the RS-CANFD module and the TDCO[6:0] value. (This value is rounded off to the nearest integer of Tq.) Usually, the TDCO[6:0] value must be equal to SS + TSEG1, the sample point timing.

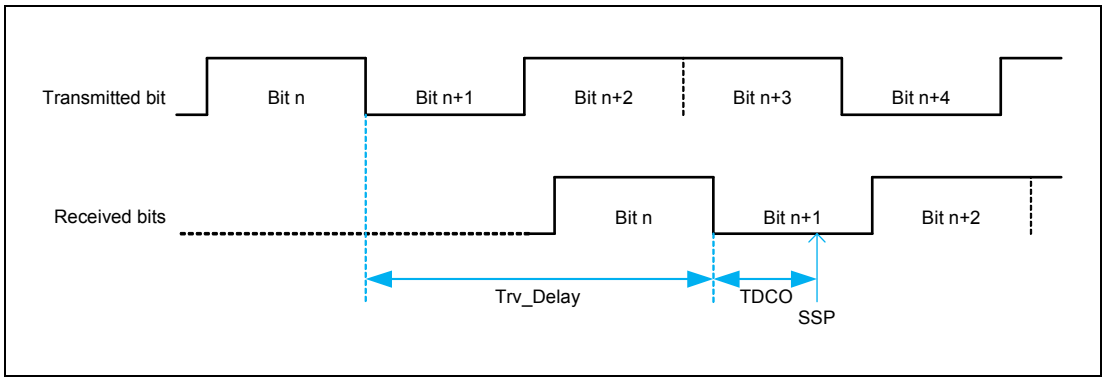



**Figure 21.22 SSP timing**

When the TDCOC bit is 1, the SSP timing is determined only by the TDCO[6:0] value. (When the DBRP[7:0] value in the RSCFDnCFDCmDCFG register is larger than 0, the TDCO[6:0] value is also rounded off to the nearest integer of Tq.)

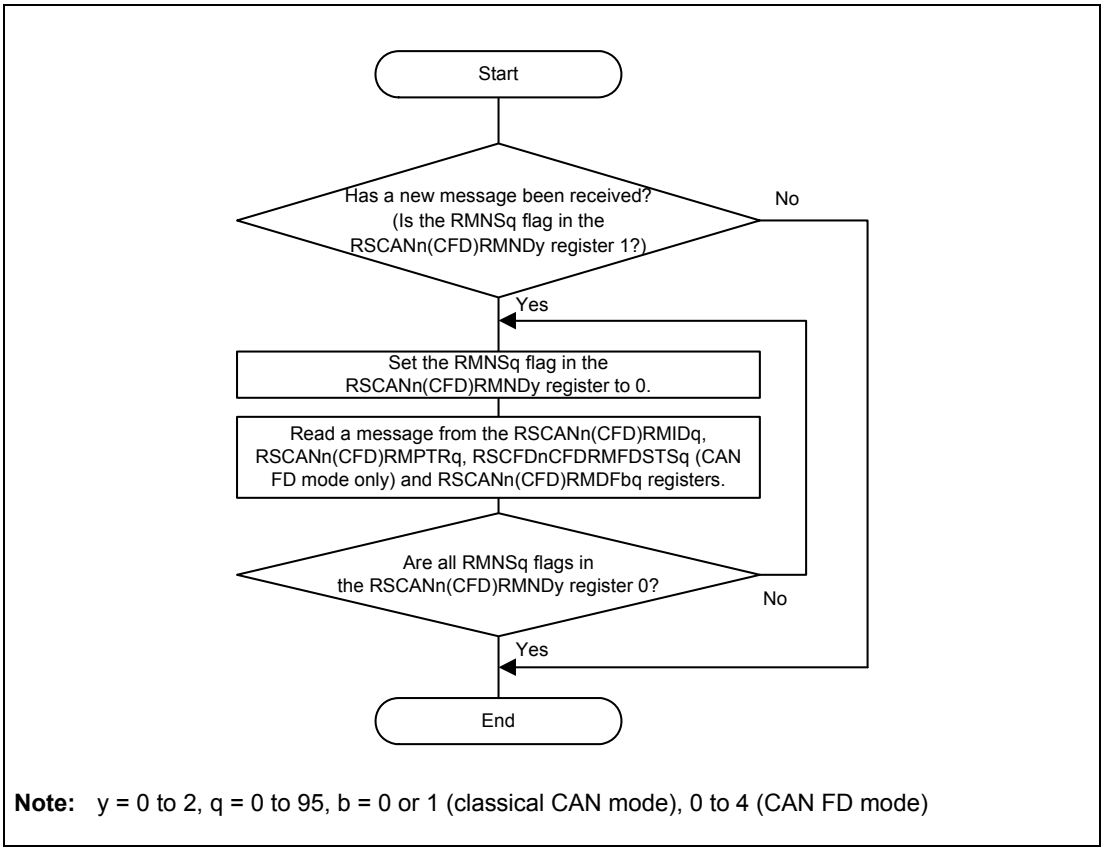
The RS-CANFD module compensates a delay up to (3 CANm bit time - 2 Tq). (CANm bit time is data bit rate values.)

When the TDCE bit is set to 1 (Transmitter delay compensation is enabled) in the RSCFDnCFDCmFDCFG register, set the equal value of 1 or less to the bits NBRP[9:0] and DBRP[7:0].

## 21.11.2 Reception Procedure

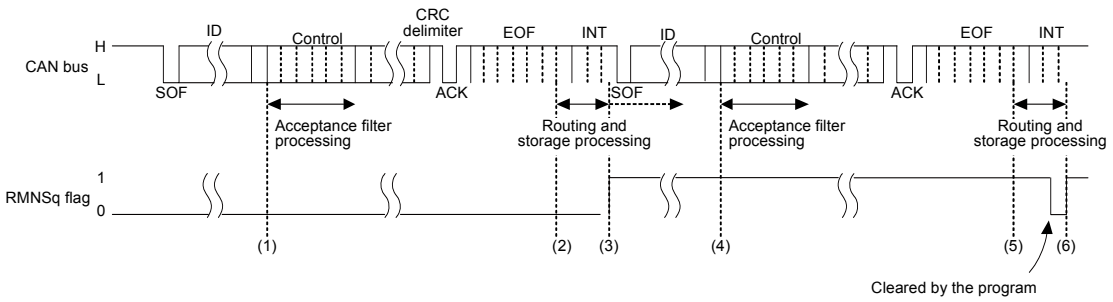
### 21.11.2.1 Receive Buffer Reading Procedure

When the processing to store received messages in a receive buffer starts, the RMNSq flag in the RSCANn(CFD)RMNDy register (y = 0 to 2, q = 0 to 95) is set to 1 (receive buffer q contains a new message). Messages can be read from registers RSCANn(CFD)RMIDq, RSCANn(CFD)RMPTRq, RSCFDnCFDRMFDSTSq (only in CAN FD mode), and RSCANn(CFD)RMDFbq (b = 0 or 1 in classical CAN mode, b = 0 to 4 in CAN FD mode). **Figure 21.23** shows the receive buffer reading procedure. This procedure ensures the consistency of messages read from registers RSCANn(CFD)RMIDq, RSCANn(CFD)RMPTRq, RSCFDnCFDRMFDSTSq, and RSCANn(CFD)RMDFbq.




**Note:** y = 0 to 2, q = 0 to 95, b = 0 or 1 (classical CAN mode), 0 to 4 (CAN FD mode)

**Figure 21.23 Receive Buffer Reading Procedure**




**Note:** y = 0 to 2, q = 0 to 95
RMNSq: Flag in the RSCANn(CFD)RMNDy register

**Figure 21.24 Receive Buffer Reception Timing Chart**

(1) When the ID field in a message has been received, the acceptance filter processing starts.

(2) When the message matches the receive rule of the corresponding channel and the message has been successfully received, the routing processing to transfer the message to the specified buffer starts. When the DCE bit in the RSCANn(CFD)GCFG register is set to 1 (DLC check is enabled), the DLC filter processing starts at this time.

(3) When the message has passed through the DLC filter processing, the processing to store the message in the specified receive buffer starts.
When the message storage processing starts, the RMNSq flag in the corresponding RSCANn(CFD)RMNDy register is set to 1 (the receive buffer contains a new message). If other channels are performing filter processing or transmit priority determination processing, the routing processing and the storage processing may be delayed.

(4) When the ID field of the next message has been received, the acceptance filter processing starts.

(5) When the message matches the receive rule of the corresponding channel and the message has been successfully received, the routing processing to transfer the message to the specified buffer starts. When the DCE bit in the RSCANn(CFD)GCFG register is set to 1 (DLC check is enabled), the DLC filter processing starts at this time.

(6) When the corresponding RMNSq flag is cleared to 0 (the receive buffer contains no new message), this flag is set to 1 again when the message storage processing starts. Even if the RMNSq flag remains 1, a new message is overwritten to the receive buffer. The RMNSq flag should not be cleared to 0 during storage of messages.

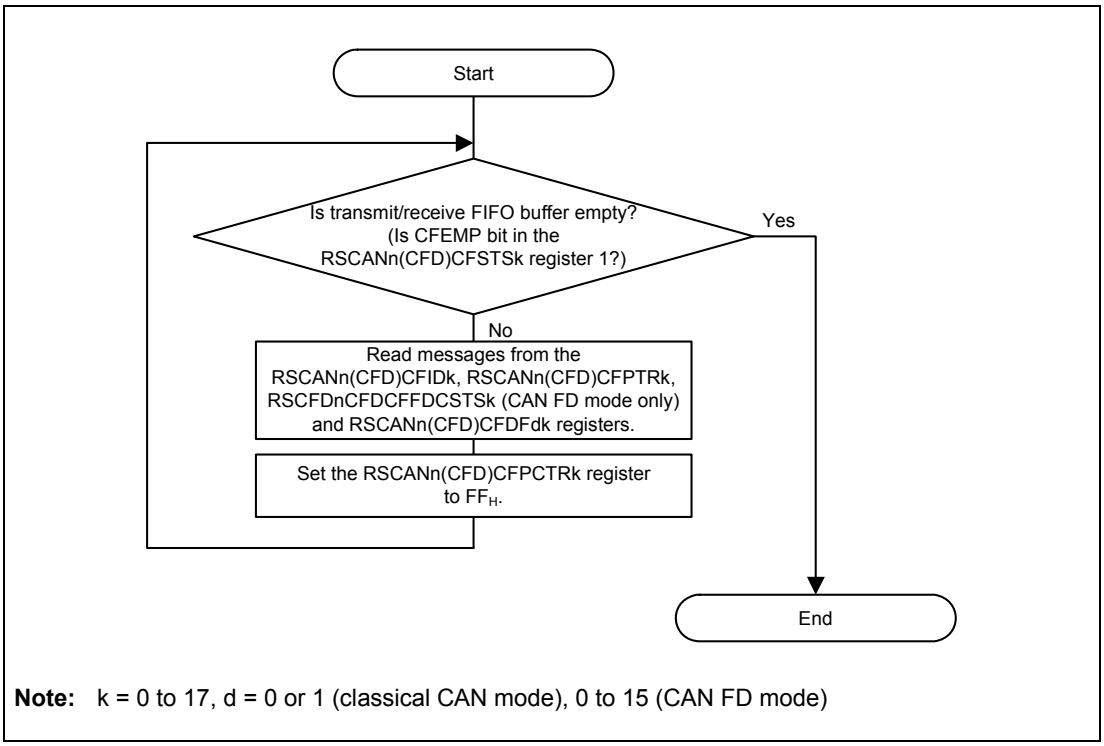
### 21.11.2.2 FIFO Buffer Reading Procedure

When received messages have been stored in one or more receive FIFO buffers or a transmit/receive FIFO buffer that is set to receive mode or gateway mode, the corresponding message count display counter (RFMC[7:0] bits in the RSCANn(CFD)RFSTSx register (x = 0 to 7) or CFMC[7:0] bits in the RSCANn(CFD)CFSTSk register (k = 0 to 17)) is incremented. At this time, when the RFIE bit (receive FIFO interrupt is enabled) in the RSCANn(CFD)RFCCx register or the CFRXIE bit (transmit/receive FIFO receive interrupt is enabled) in the RSCANn(CFD)CFCCk register is set to 1, an interrupt request is generated. Received messages can be read from the RSCANn(CFD)RFIDx, RSCANn(CFD)RFPTRx, RSCFDnCFDRFFDSTSx (only in CAN FD mode), and RSCANn(CFD)RFDFdx (d = 0 or 1 in classical CAN mode, d = 0 to 15 in CAN FD mode) registers for receive FIFO buffers, or from the RSCANn(CFD)CFIDk, RSCANn(CFD)CFPTRk, RSCFDnCFDCFFDCSTSk (only in CAN FD mode), and RSCANn(CFD)CFDFdk registers for transmit/receive FIFO buffers. Messages in FIFO buffers can be read sequentially on a first-in, first-out basis.

When the message count display counter value matches the FIFO buffer depth (a value set by the RFDC[2:0] bits in the RSCANn(CFD)RFCCx register or the CFDC[2:0] bits in the RSCANn(CFD)CFCCk register), the RFFLL or CFFLL flag is set to 1 (the receive FIFO buffer is full).

When all messages have been read out of the FIFO buffer, the RFEMP flag in the RSCANn(CFD)RFSTSx register or the CFEMP flag in the RSCANn(CFD)CFSTSk register is set to 1 (the receive FIFO buffer contains no unread message (buffer empty)).

If the RFE bit or the CFE bit is cleared to 0 (no receive FIFO buffer is used) with the interrupt request flag (RFIF flag in the RSCANn(CFD)RFSTSx register or CFRXIF flag in the RSCANn(CFD)CFSTSk register) set to 1 (a receive FIFO interrupt request is present), the interrupt request flag is not automatically cleared to 0. The program must clear the interrupt request flag to 0.




**Note:** k = 0 to 17, d = 0 or 1 (classical CAN mode), 0 to 15 (CAN FD mode)

**Figure 21.25 Transmit/Receive FIFO Buffer Reading Procedure**

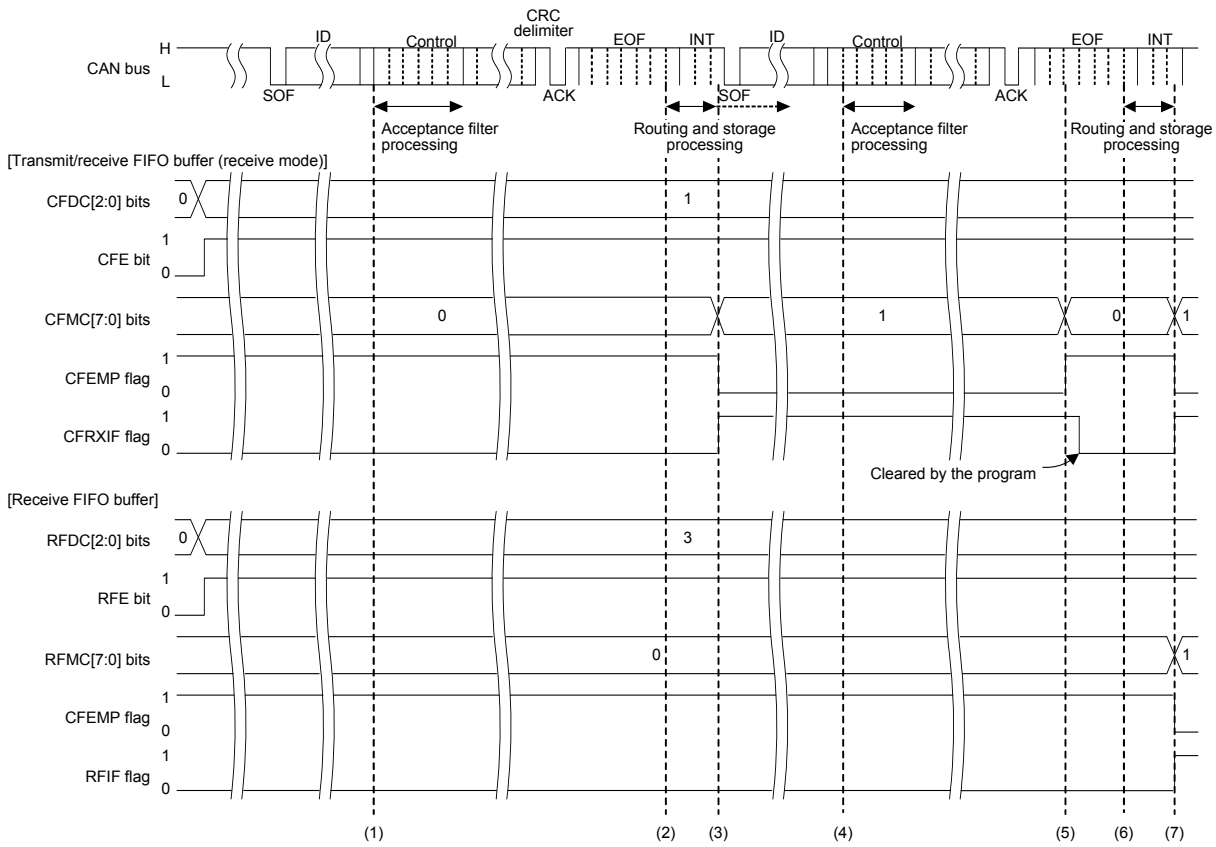
When reading a message in CAN FD mode, do not read the RSCFDnCFDRFDFd_x or RSCFDnCFDCFDFd_k register corresponding to the area exceeding the payload storage size specified by the RFPLS[2:0] bits in the RSCFDnCFDRFCCx register or the CFPLS[2:0] bits in the RSCFDnCFDCFCCk register.

**Table 21.188 Payload Storage Area of Receive FIFO Buffer**

| RFPLS[2:0] Setting Value | Payload Storage Size | Corresponding Data Field Registers |
|---|---|---|
| $000_B$ | 8 bytes | RSCFDnCFDRFDF0_x to RSCFDnCFDRFDF1_x |
| $001_B$ | 12 bytes | RSCFDnCFDRFDF0_x to RSCFDnCFDRFDF2_x |
| $010_B$ | 16 bytes | RSCFDnCFDRFDF0_x to RSCFDnCFDRFDF3_x |
| $011_B$ | 20 bytes | RSCFDnCFDRFDF0_x to RSCFDnCFDRFDF4_x |
| $100_B$ | 24 bytes | RSCFDnCFDRFDF0_x to RSCFDnCFDRFDF5_x |
| $101_B$ | 32 bytes | RSCFDnCFDRFDF0_x to RSCFDnCFDRFDF7_x |
| $110_B$ | 48 bytes | RSCFDnCFDRFDF0_x to RSCFDnCFDRFDF11_x |
| $111_B$ | 64 bytes | RSCFDnCFDRFDF0_x to RSCFDnCFDRFDF15_x |

**Table 21.189 Payload Storage Area of Transmit/Receive FIFO Buffer**

| CFPLS[2:0] Setting Value | Payload Storage Size | Corresponding Data Field Registers |
|---|---|---|
| $000_B$ | 8 bytes | RSCFDnCFDCFDF0_k to RSCFDnCFDCFDF1_k |
| $001_B$ | 12 bytes | RSCFDnCFDCFDF0_k to RSCFDnCFDCFDF2_k |
| $010_B$ | 16 bytes | RSCFDnCFDCFDF0_k to RSCFDnCFDCFDF3_k |
| $011_B$ | 20 bytes | RSCFDnCFDCFDF0_k to RSCFDnCFDCFDF4_k |
| $100_B$ | 24 bytes | RSCFDnCFDCFDF0_k to RSCFDnCFDCFDF5_k |
| $101_B$ | 32 bytes | RSCFDnCFDCFDF0_k to RSCFDnCFDCFDF7_k |
| $110_B$ | 48 bytes | RSCFDnCFDCFDF0_k to RSCFDnCFDCFDF11_k |
| $111_B$ | 64 bytes | RSCFDnCFDCFDF0_k to RSCFDnCFDCFDF15_k |




CFDC[2:0], CFE: Bits in the RSCANn(CFD)CFCCk register
CFMC[7:0], CFEMP, CFRXIF: Flags in the RSCANn(CFD)CFSTSk register
RFDC[2:0], RFE: Bits in the RSCANn(CFD)RFCCx register
RFMC[7:0], CFEMP, RFIF: Flags in the RSCANn(CFD)RFSTSx register

**Note:** k = 0 to 17, x = 0 to 7

**Figure 21.26 FIFO Buffer Reception Timing Chart**

(1) When the ID field in a message has been received, the acceptance filter processing starts.

(2) When the message matches the receive rule of the corresponding channel and the message has been successfully received, the routing processing to transfer the message to the specified buffer starts. When the DCE bit in the RSCANn(CFD)GCFG register is set to 1 (DLC check is enabled), the DLC filter processing starts at this time.

(3) When the message has passed through the DLC filter processing and the CFE bit in the RSCANn(CFD)CFCCk register is 1 (transmit/receive FIFO buffers are used) and the CFDC[2:0] value in the RSCANn(CFD)CFCCk register is $001_B$ or more, the message is stored in the transmit/receive FIFO buffer that is set to receive mode. The CFMC[7:0] value in the RSCANn(CFD)CFSTSk register is incremented and becomes $01_H$. When the CFIM bit in the RSCANn(CFD)CFCCk register is set to 1 (a FIFO receive interrupt request is generated each time a message has been received), the CFRXIF flag in the RSCANn(CFD)CFSTSk register is set to 1 (a transmit/receive FIFO receive interrupt request is present). The CFRXIF flag can be reset to 0 by the program.

(4) When the ID field of the next message has been received, the acceptance filter processing starts.

(5) Read received messages from the RSCANn(CFD)CFIDk, RSCANn(CFD)CFPTRk, and RSCANn(CFD)CFDFdk registers and write $FF_H$ to the RSCANn(CFD)CFPCTRk register. This causes the CFMC[7:0] bits in the RSCANn(CFD)CFSTSk register to be decremented. When CFMC[7:0] becomes $00_H$, the CFEMP flag in the RSCANn(CFD)CFSTSk register becomes 1 (the transmit/receive FIFO buffer contains no message (buffer empty)).

(6) When the message matches the receive rule of the corresponding channel and the message has been successfully received, the routing processing to transfer the message to the specified buffer starts. When the DCE bit in the RSCANn(CFD)GCFG register is set to 1 (DLC check is enabled), the DLC filter processing starts at this time.

(7) The message is stored in the transmit/receive FIFO buffer set in receive mode when the message has passed through the DLC filter process if the CFE bit is set to 1 (transmit/receive FIFO buffers are used), and the CFDC[2:0] bits are set to $001_B$ or more. The CFMC[7:0] bit value is incremented by 1 to be $01_H$. When the CFIM bit is set to 1 (an interrupt occurs each time a message has been received), the CFRXIF flag is set to 1 (a transmit/receive FIFO receive interrupt request is present).
The message is stored in the receive FIFO buffer if the RFE bit in the RSCANn(CFD)RFCCx register is set to 1 (receive FIFO buffers are used), and the RFDC[2:0] bits in the RSCANn(CFD)RFCCx register are set to $001_B$ or more. The RFMC[7:0] bits in the RSCANn(CFD)RFSTSx register are set to $01_H$ by being incremented by 1. When the RFIM bit in the RSCANn(CFD)RFCCx register is set to 1 (an interrupt occurs each time a message has been received), the RFIF flag in the RSCANn(CFD)RFSTSx register is set to 1 (a receive FIFO interrupt request is present).

### 21.11.2.3 FIFO Buffer Reading Procedure by DMA Transfer

In CAN FD mode, the following FIFO buffers can be read by DMA transfer.

- All receive FIFO buffers x (x = 0 to 7)
- The first transmit/receive FIFO buffer k allocated to channel m (k = 3 × m, m = 0 to 5)

The DMA enable bit (RFDMAEx or CFDMAEm bit in the RSCFDnCFDCDTC register) can be set at any time. However, before setting the DMA enable bit to 1 (to enable DMA transfer requests), set the receive interrupt enable bit (RFIE bit in the RSCFDnCFDRFCCx register or CFRXIE bit in the RSCFDnCFDCFCCk register) of related FIFO buffers to 0 (to disable interrupts). When DMA transfer requests are enabled, do not write a value to the FIFO control register (RSCFDnCFDRFCCx register or RSCFDnCFDCFCCk register).

When an unread message is remaining in a DMA transfer-enabled FIFO buffer, a DMA transfer request trigger is generated. Specify the FIFO access register address for the transfer source address, and adjust the transfer size so that data can be read to the end of the payload storage area with a single trigger. The end of the payload storage area depends on the payload storage size specified by the RFPLS[2:0] bits in the RSCFDnCFDRFCCx register or the CFPLS[2:0] bits in the RSCFDnCFDCFCCk register.

After the end of the payload stored in the FIFO buffer has been read, the RFMC[7:0] value in the RSCFDnCFDRFSTSx register or the CFMC[7:0] value in the RSCFDnCFDCFSTSk register is automatically decremented. After that, if an unread message is remaining in the FIFO buffer, a trigger is generated again.

When the RFDMAEx or CFDMAEm bit is set to 0 (to disable DMA transfer requests) during DMA transfer, wait until the DMA transfer status (RFDMASTSx or CFDMASTSm bit in the RSCFDnCFDCDTSTS register) is cleared to 0 (not in DMA transfer processing), and then start the next processing (enabling DMA transfer again etc.). When disabling DMA transfer, examine how to process a message remaining in the FIFO buffer and a newly arriving message. When the FIFO buffer is enabled, it continues to receive messages.

## 21.11.3 Transmission Procedure

### 21.11.3.1 Procedure for Transmission from Transmit Buffers

**Figure 21.27** shows the procedure for transmission from transmit buffers.

**Figure 21.28** shows a timing chart where messages are transmitted from two transmit buffers in the same channel and transmission has been successfully completed. **Figure 21.29** shows a timing chart where messages are transmitted from two transmit buffers in the same channel and transmit abort has been completed.

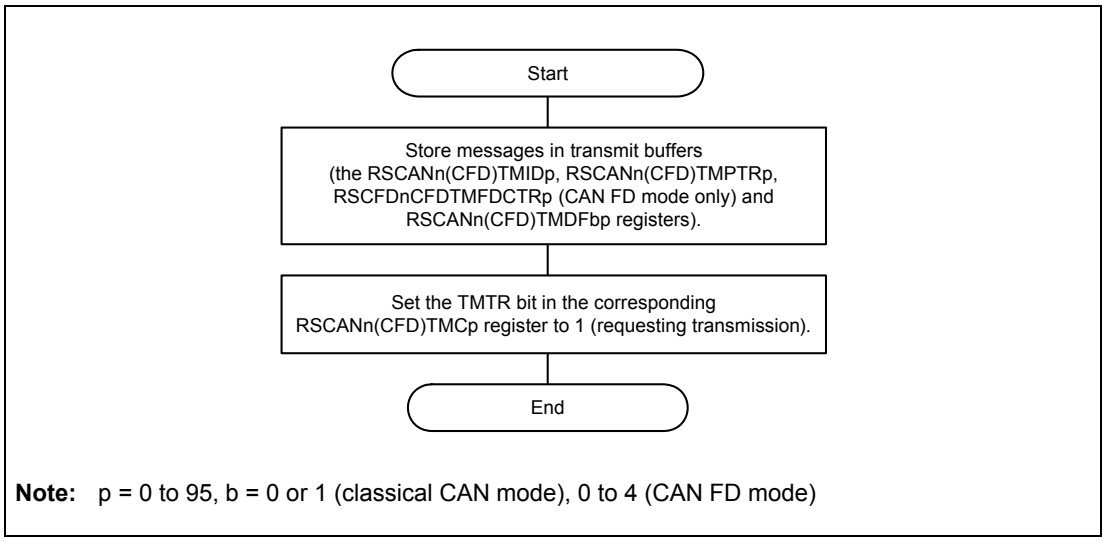
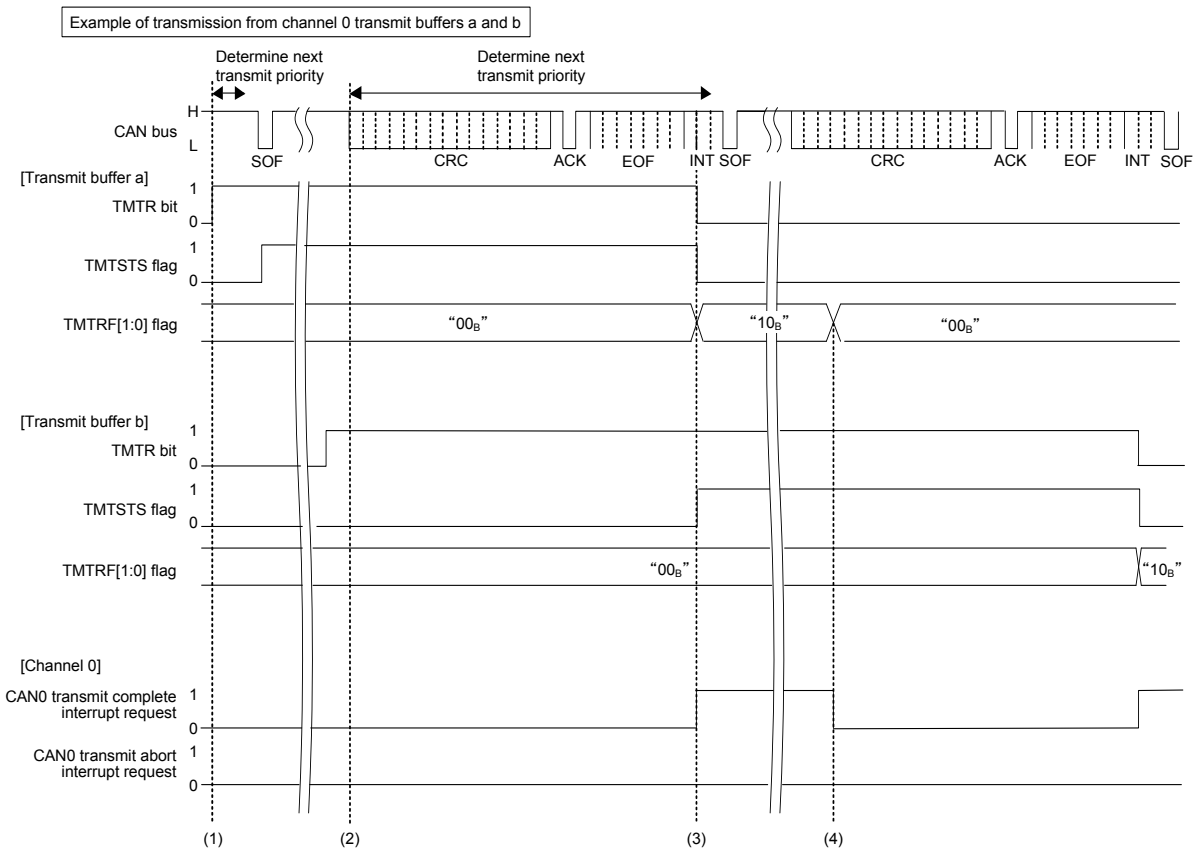



**Figure 21.27 Procedure for Transmission from Transmit Buffers**

In CAN FD mode and transmit buffer merge mode, messages with a payload size of more than 20 bytes can be transmitted from transmit buffers (16 × m) + 0 and transmit buffers (16 × m) + 3. At this time, transmit buffers (16 × m) + 1 to (16 × m) + 2 and transmit buffers (16 × m) + 4 to (16 × m) + 5 are allocated as a payload storage area. Registers RSCFDnCFDTMIDp, RSCFDnCFDTMPTRp, and RSCFDnCFDTMFDCTRp corresponding to these buffers can be used as data field registers that can store 4-byte data bytes (payload) like the RSCFDnCFDTMDFb_p register. **Table 21.190** shows message storage registers when transmitting a message with a payload size of more than 20 bytes from transmit buffer 0.

**Table 21.190 Message Storage Registers in Transmit Buffer Merge Mode (Example of Transmit Buffer 0)**

| Transmit Buffer | Offset from Base Address | Symbol | Register Function in Transmit Buffer Merge Mode |
|---|---|---|---|
| Transmit buffer 0 | $4000_H$ | RSCFDnCFDTMID0 | Transmit buffer 0 ID data, transmit history data store enable bit, RTR bit,and IDE bit |
| | $4004_H$ | RSCFDnCFDTMPTR0 | Transmit buffer 0 label data and DLC data |
| | $4008_H$ | RSCFDnCFDTMFDCTR0 | Transmit buffer 0 ESI bit, BRS bit, and FDF bit |
| | $400C_H$ to $401C_H$ | RSCFDnCFDTMDF0_0 to RSCFDnCFDTMDF4_0 | Transmit buffer 0 data bytes 0, 1, 2, and 3 to transmit buffer 0 data bytes 16, 17, 18, and 19 |
| Transmit buffer 1 | $4020_H$ | RSCFDnCFDTMID1 | Transmit buffer 0 data bytes 20, 21, 22, and 23 |
| | $4024_H$ | RSCFDnCFDTMPTR1 | Transmit buffer 0 data bytes 24, 25, 26, and 27 |
| | $4028_H$ | RSCFDnCFDTMFDCTR1 | Transmit buffer 0 data bytes 28, 29, 30, and 31 |
| | $402C_H$ to $403C_H$ | RSCFDnCFDTMDF0_1 to RSCFDnCFDTMDF4_1 | Transmit buffer 0 data bytes 32, 33, 34, and 35 to transmit buffer 0 data bytes 48, 49, 50, and 51 |
| Transmit buffer 2 | $4040_H$ | RSCFDnCFDTMID2 | Transmit buffer 0 data bytes 52, 53, 54, and 55 |
| | $4044_H$ | RSCFDnCFDTMPTR2 | Transmit buffer 0 data bytes 56, 57, 58, and 59 |
| | $4048_H$ | RSCFDnCFDTMFDCTR2 | Transmit buffer 0 data bytes 60, 61, 62, and 63 |
| | $404C_H$ to $405C_H$ | RSCFDnCFDTMDF0_2 to RSCFDnCFDTMDF4_2 | Not used |




**Note:** a = 0 to 15, b = 0 to 15
TMTR: Bit in the RSCANn(CFD)TMCp register
TMTSTS, TMTRF[1:0]: Flags in the RSCANn(CFD)TMSTSp registers

**Figure 21.28 Transmit Buffer Transmission Timing Chart (Transmission Completed Successfully)**

(1) When the TMTR bit in the RSCANn(CFD)TMCa register is set to 1 while the CAN bus is idle, the transmit priority determination processing starts to determine the highest-priority transmit buffer. If transmit buffer a is determined to be the highest-priority transmit buffer, the TMTSTS flag in the corresponding RSCANn(CFD)TMSTSa register is set to 1 (transmission is in progress) and the CAN channel starts transmitting data.

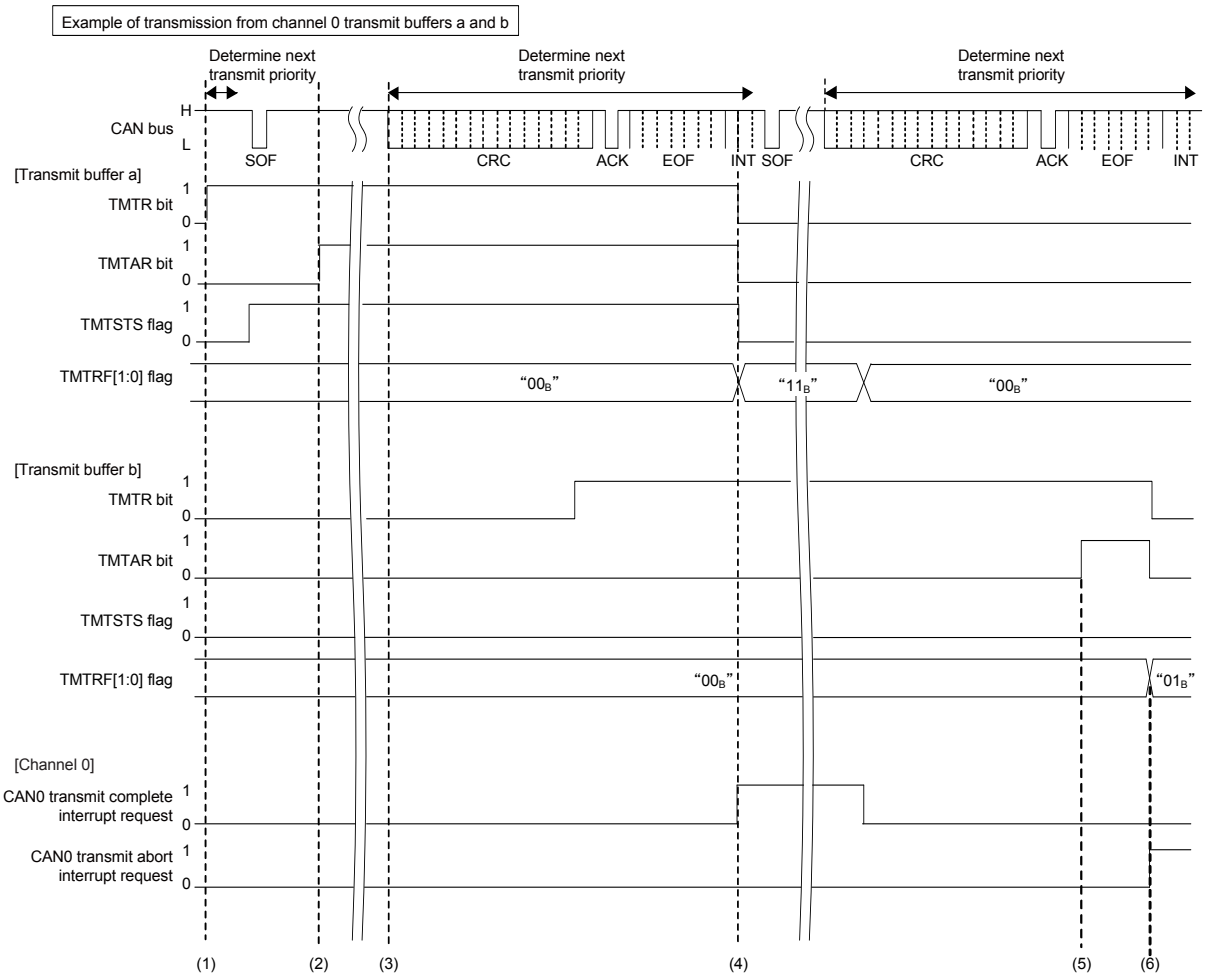
(2) When a transmit request from a buffer is present, the priority determination starts at the first bit of CRC field for the next transmission. The determination time may delay if the transmit priority determination processing is performed on another channel. However, the delay does not occur during transmission because the determination processing is completed by the third bit of the intermission.

(3) When transmission completes successfully, the TMTRF[1:0] flag in the RSCANn(CFD)TMSTSa register is set to $10_B$ (transmission has been completed (without transmit abort request)) and the TMTSTS flag and the TMTR bit in the RSCANn(CFD)TMCa register are cleared to 0. When the TMIEa bit in the RSCANn(CFD)TMIEC0 register is 1 (transmit buffer interrupt is enabled), a CAN0 transmit complete interrupt request is generated. To clear the interrupt request, set the TMTRF[1:0] flag to $00_B$ (transmission is in progress or no transmit request is present).

(4) Before starting the next transmission, set the TMTRF[1:0] flag to $00_B$. Write the next message to the transmit buffer, and then set the TMTR bit to 1 (transmission is requested). The TMTR bit can be set to 1 only when the TMTRF[1:0] flag value is $00_B$.

If an arbitration-lost has occurred after transmission is started, the TMTSTS flag is cleared to 0. The transmit priority determination is reexecuted at the beginning of the CRC field to search the highest-priority transmit buffer. If an error has occurred during transmission or after arbitration loss, the priority determination processing is reexecuted during transmission of an error frame.

When a 2-bit ECC error is detected during the priority determination processing, no data is transmitted (in classical CAN mode, only when the EEFE bit in the RSCANnGCFG register is 1).




**Note:** a = 0 to 15, b = 0 to 15
TMTR, TMTAR: Bits in the RSCANn(CFD)TMCp register
TMTSTS, TMTRF[1:0]: Flags in the RSCANn(CFD)TMSTSp register

**Figure 21.29 Transmit Buffer Transmission Timing Chart (Transmit Abort Completed)**

(1) When the TMTR bit in the RSCANn(CFD)TMCa register is set to 1 while the CAN bus is idle, the transmit priority determination processing starts to determine the highest-priority transmit buffer. If transmit buffer a is determined to be the highest-priority transmit buffer, the TMTSTS flag in the corresponding RSCANn(CFD)TMSTSa register is set to 1 (transmission is in progress) and the CAN channel starts transmitting data.

(2) When it is determined that the transmit buffer is used for the next transmission or transmission is in progress, message transmission is not aborted unless an error or arbitration loss occurs even if the TMTAR bit is set to 1 (transmit abort is requested).

(3) The priority determination starts at the first bit of the CRC field for the next transmission. In this timing chart, buffer b is not selected as the next transmit buffer. The determination time may delay if the transmit priority determination processing is performed on another channel. However, the delay does not occur during transmission because the determination processing is completed by the third bit of the intermission.

(4) When transmission completes successfully, the TMTRF[1:0] flag in the RSCANn(CFD)TMSTSa register is set to $11_B$ (transmission has been completed (with transmit abort request)) and the TMTSTS flag and the TMTR bit in the RSCANn(CFD)TMCa register are cleared to 0. When the TMIEa value in the RSCANn(CFD)TMIEC0 register is 1 (transmit buffer interrupt is enabled), a CAN0 transmit complete interrupt request is generated. To clear the interrupt request, set the TMTRF[1:0] flag to $00_B$ (transmission is in progress or no transmit request is present).

(5) While another CAN node is transmitting data on the CAN bus (TMTSTS flag = 0), if the TMTAR bit is set to 1 while the corresponding channel is determining transmit priority, the TMTR bit cannot be cleared to 0.

(6) After the internal processing time has passed, the transmission is terminated and the TMTRF[1:0] flag is set to $01_B$. When the transmit buffer is not transmitting data and is not selected as the next transmit buffer and priority determination is not being made, an abort request is immediately accepted and the TMTRF[1:0] flag is set to $01_B$. At this time, the TMTR and TMTAR bits are cleared to 0. When transmit abort is completed with the TAIE bit in the RSCANn(CFD)CmCTR register set to 1 (transmit abort interrupt is enabled), an interrupt request is generated. To clear the interrupt request, set the TMTRF[1:0] flag to $00_B$.

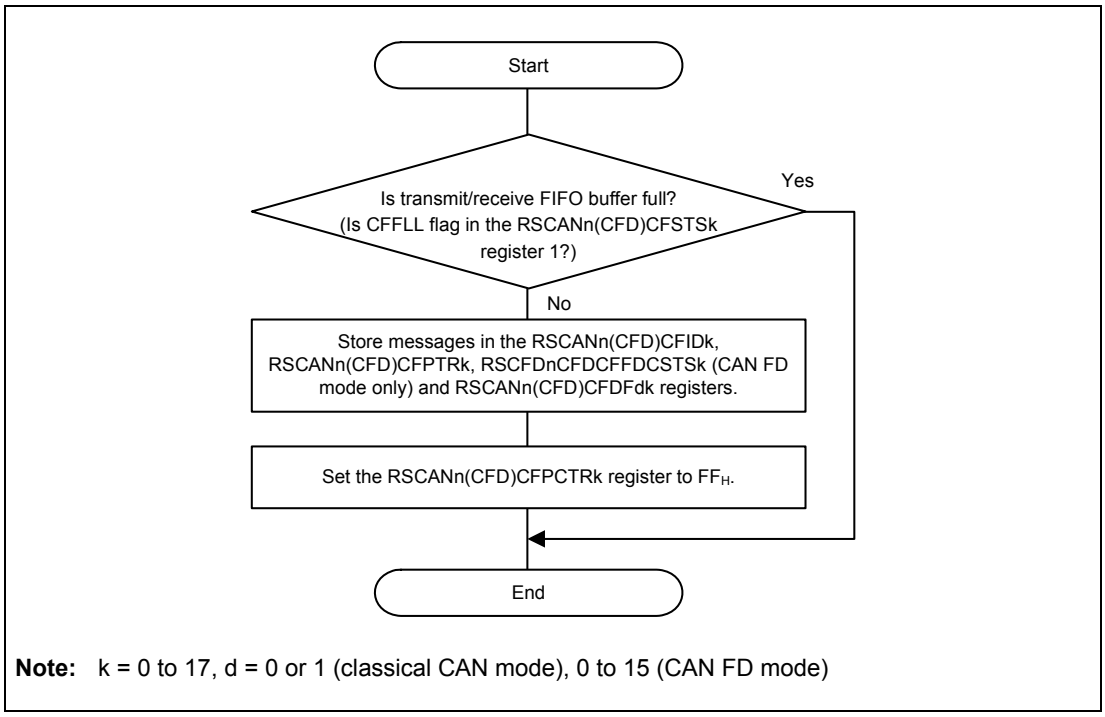
If an arbitration loss has occurred after the CAN channel started transmission, the TMTSTS bit is cleared to 0. The transmit priority determination is reexecuted at the beginning of the CRC field to find the highest-priority transmit buffer. If an error has occurred during transmission or after arbitration loss, the priority determination processing is reexecuted during transmission of an error frame.

When a 2-bit ECC error is detected during the priority determination processing, no data is transmitted (in classical CAN mode, only when the EEFE bit in the RSCANnGCFG register is 1).

### 21.11.3.2 Procedure for Transmission from Transmit/Receive FIFO Buffers

**Figure 21.30** shows the procedure for transmission from transmit/receive FIFO buffers.

**Figure 21.31** shows a timing chart where messages are transmitted from two transmit/receive FIFO buffers in the same channel and transmission has been successfully completed. **Figure 21.32** shows a timing chart where messages are transmitted from two transmit/receive FIFO buffers in the same channel and transmit abort has been completed.



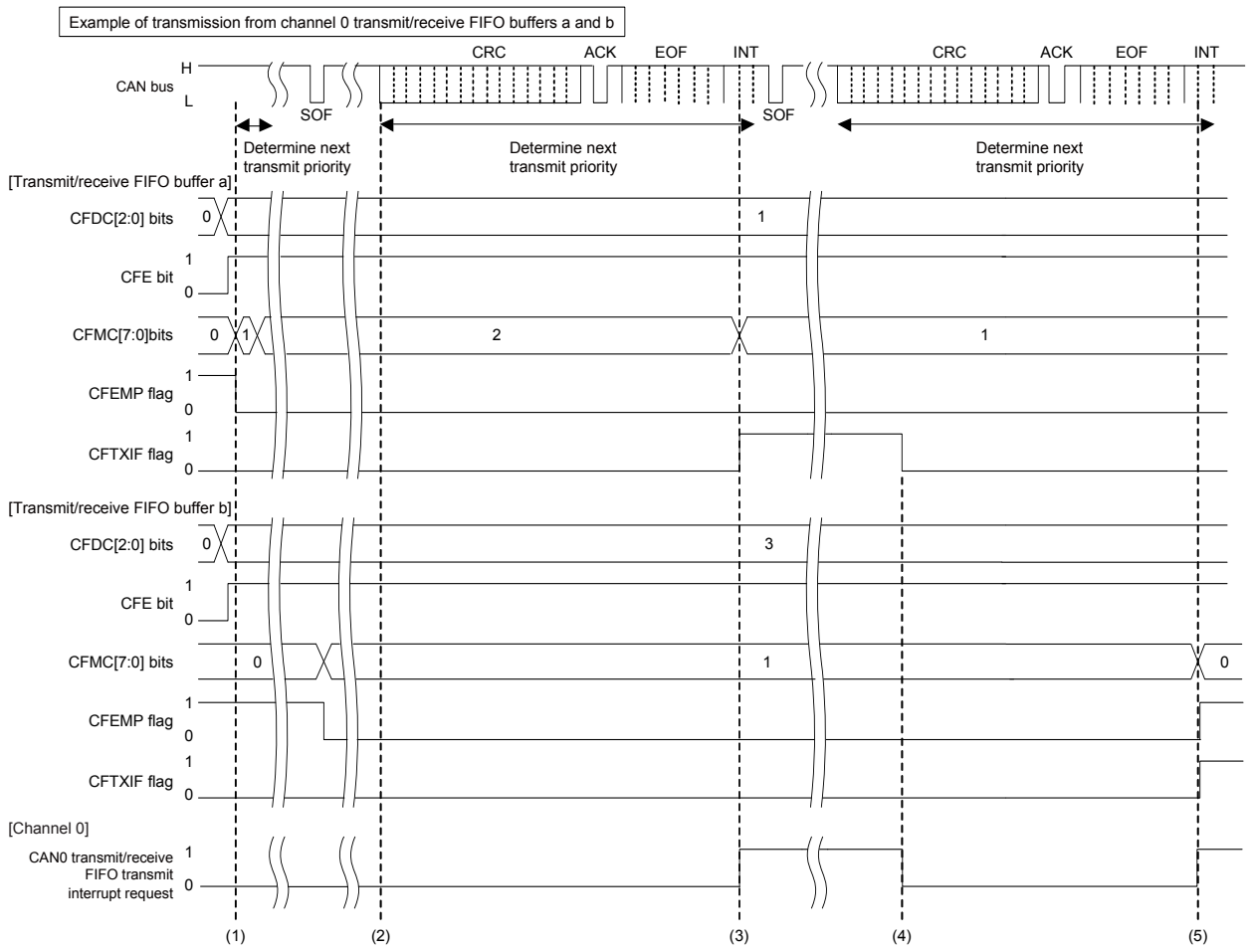

**Note:** k = 0 to 17, d = 0 or 1 (classical CAN mode), 0 to 15 (CAN FD mode)

**Figure 21.30 Procedure for Transmission from Transmit/Receive FIFO Buffers**

When storing a message, do not write a value to the RSCFDnCFDCFDFd_k register corresponding to the area exceeding the payload storage size specified by the CFPLS[2:0] bits in the RSCFDnCFDCFCCk register.

**Table 21.191 Payload Storage Area of Transmit/Receive FIFO Buffer**

| CFPLS[2:0] Setting Value | Payload Storage Size | Corresponding Data Field Registers |
|---|---|---|
| $000_B$ | 8 bytes | RSCFDnCFDCFDF0_k to RSCFDnCFDCFDF1_k |
| $001_B$ | 12 bytes | RSCFDnCFDCFDF0_k to RSCFDnCFDCFDF2_k |
| $010_B$ | 16 bytes | RSCFDnCFDCFDF0_k to RSCFDnCFDCFDF3_k |
| $011_B$ | 20 bytes | RSCFDnCFDCFDF0_k to RSCFDnCFDCFDF4_k |
| $100_B$ | 24 bytes | RSCFDnCFDCFDF0_k to RSCFDnCFDCFDF5_k |
| $101_B$ | 32 bytes | RSCFDnCFDCFDF0_k to RSCFDnCFDCFDF7_k |
| $110_B$ | 48 bytes | RSCFDnCFDCFDF0_k to RSCFDnCFDCFDF11_k |
| $111_B$ | 64 bytes | RSCFDnCFDCFDF0_k to RSCFDnCFDCFDF15_k |




**Note:** a = 0 to 2, b = 0 to 2
CFDC[2:0],CFE: Bits in the RSCANn(CFD)CFCCk register
CFMC[7:0], CFEMP, CFTXIF: Flags in the RSCANn(CFD)CFSTSk register

**Figure 21.31 Transmit/Receive FIFO Buffer Transmission Timing Chart (Transmission Completed Successfully)**

(1) While the CAN bus is idle, when the CFE bit in the RSCANn(CFD)CFCCa register is 1 (transmit/receive FIFO buffers are used) and the CFDC[2:0] value in the RSCANn(CFD)CFCCa register is $001_B$ (4 messages) or more and the CFMC[7:0] value in the RSCANn(CFD)CFSTSa register is $01_H$ or more, the priority determination processing starts to determine the highest-priority transmit message. When the highest-priority transmit message has been determined, transmission of the message starts. In this figure, the message is transmitted from transmit/receive FIFO buffer a of channel 0.

(2) When a transmit request from a buffer is present, the priority determination starts at the first bit of the CRC field for the next transmission. The determination time may delay if the transmit priority determination processing is performed on another channel. However, the delay does not occur during transmission because the determination processing is completed by the third bit of the intermission.

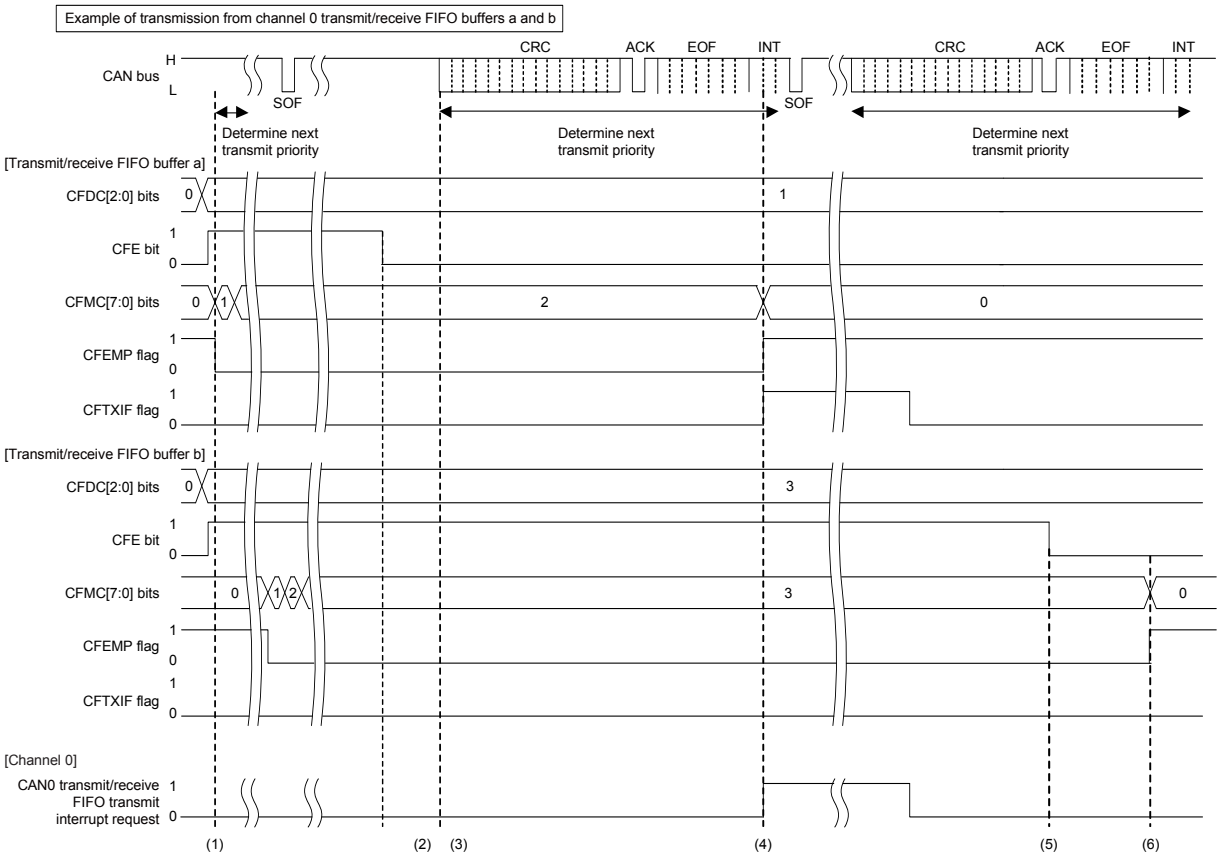
(3) When transmission completes successfully, the CFMC[7:0] value in the RSCANn(CFD)CFSTSa register is decremented. Setting the CFIM bit in the RSCANn(CFD)CFCCa register to 1 (a FIFO

transmit interrupt request is generated each time a message has been transmitted) sets the CFTXIF flag in the RSCANn(CFD)CFSTSk register to 1 (a transmit/receive FIFO transmit interrupt request is present).

(4) The program can clear the CFTXIF flag.

(5) Message transmission from transmit/receive FIFO buffer b of channel 0 has been completed and the CFMC[7:0] value in the RSCANn(CFD)CFSTSb register is decremented. The CFMC[7:0] bits are cleared to $00_H$ and therefore the CFEMP flag in the RSCANn(CFD)CFSTSk register is set to 1 (the transmit/receive FIFO buffer contains no message (buffer empty)).

Transmission is continued until the CFEMP flag is set to 1. It is possible to continuously store transmit messages in FIFO buffers until the CFFLL flag in the RSCANn(CFD)CFSTSa and RSCANn(CFD)CFSTSb register is set to 1 (the transmit/receive FIFO buffer is full).




**Note:** a = 0 to 2, b = 0 to 2
CFDC[2:0], CFE: Bits in the RSCANn(CFD)CFCCk register
CFMC[7:0], CFEMP, CFTXIF: Flags in the RSCANn(CFD)CFSTSk register

**Figure 21.32 Transmit/Receive FIFO Buffer Transmission Timing Chart (Transmit Abort Completed)**

(1) While the CAN bus is idle, when the CFE bit in the RSCANn(CFD)CFCCa register is 1 (transmit/receive FIFO buffers are used) and the CFDC[2:0] value in the RSCANn(CFD)CFCCa register is $001_B$ (4 messages) or more and the CFMC[7:0] value in the RSCANn(CFD)CFSTSa register is $01_H$ or more, the priority determination processing starts to determine the highest-priority transmit message. When the highest-priority transmit message has been determined, transmission of the message starts. In this figure, the message is transmitted from transmit/receive FIFO buffer a of channel 0.

(2) When transmission is in progress or it is determined that the transmit/receive FIFO buffer is used for the next transmission, message transmission is not aborted unless an error or arbitration loss occurs even if the CFE bit is set to 0 (no transmit/receive FIFO buffer is used).

(3) When a transmit request from a buffer is present, the priority determination starts at the first bit of the CRC field for the next transmission. In this figure, transmit/receive FIFO buffer b is not selected as a buffer for the next transmission. The determination time may delay if the transmit priority determination processing is performed on another channel. However, the delay does not occur during transmission because the determination processing is completed by the third bit of the intermission.

(4) When transmit completes successfully, the CFMC[7:0] value is cleared to $00_H$. Setting the CFIM bit to 1 (a FIFO transmit interrupt request is generated each time a message has been transmitted) sets the CFTXIF flag in the RSCANn(CFD)CFSTSa register to 1 (a transmit/receive FIFO transmit interrupt request is present). The program can clear the CFTXIF flag.

(5) If another CAN node on the CAN bus is transmitting data (not from transmit/receive FIFO buffer b), transmit/receive FIFO buffers cannot be disabled immediately even if the CFE bit in the RSCANn(CFD)CFCCb register is cleared to 0 (no transmit/receive FIFO buffer is used) during transmit priority determination. (The CFEMP flag in the RSCANn(CFD)CFSTSb register is not set to 1 (the transmit/receive FIFO buffer contains no message (buffer empty)) immediately.)

(6) After the internal processing time has passed, transmit/receive FIFO buffers are disabled and the CFMC[7:0] bits in the RSCANn(CFD)CFSTSb register are cleared to $00_H$ and the CFEMP flag is set to 1. When the transmit/receive FIFO buffer is not transmitting data and is not selected as the next transmit buffer and priority determination is not in progress, the transmit/receive FIFO buffer is immediately disabled. (The CFMC[7:0] bits are cleared to $00_H$ and the CFEMP flag is set to 1.)

### 21.11.3.3 Procedure for Transmission from the Transmit Queue

**Figure 21.33** shows the procedure for transmission from the transmit queue.

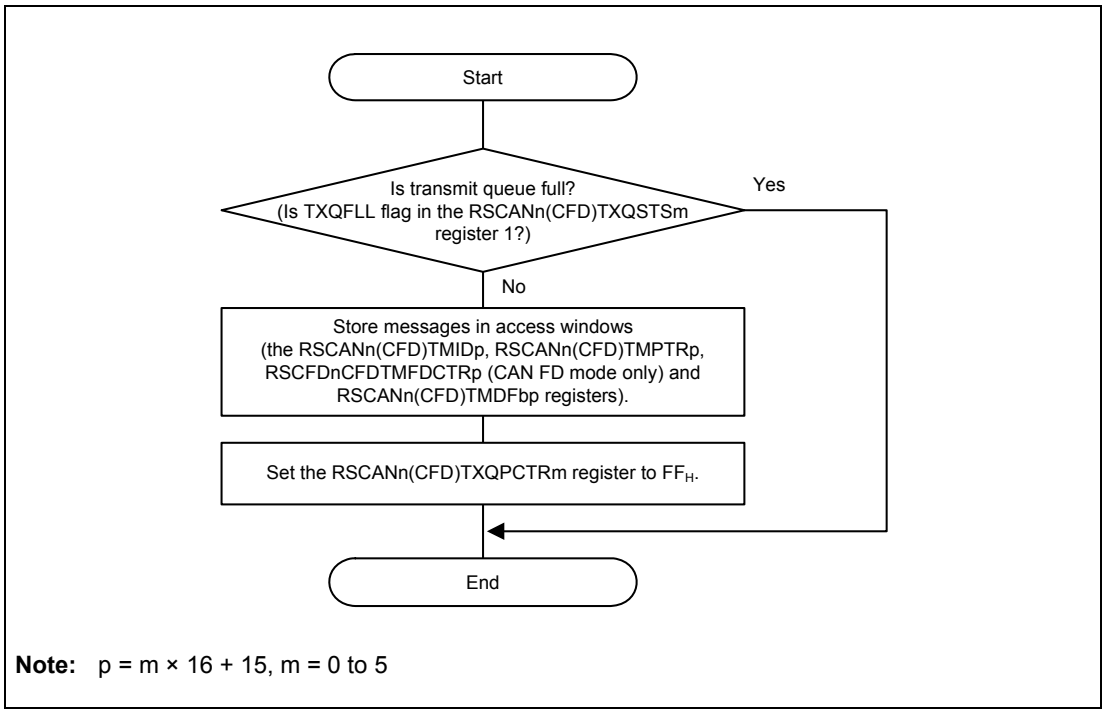
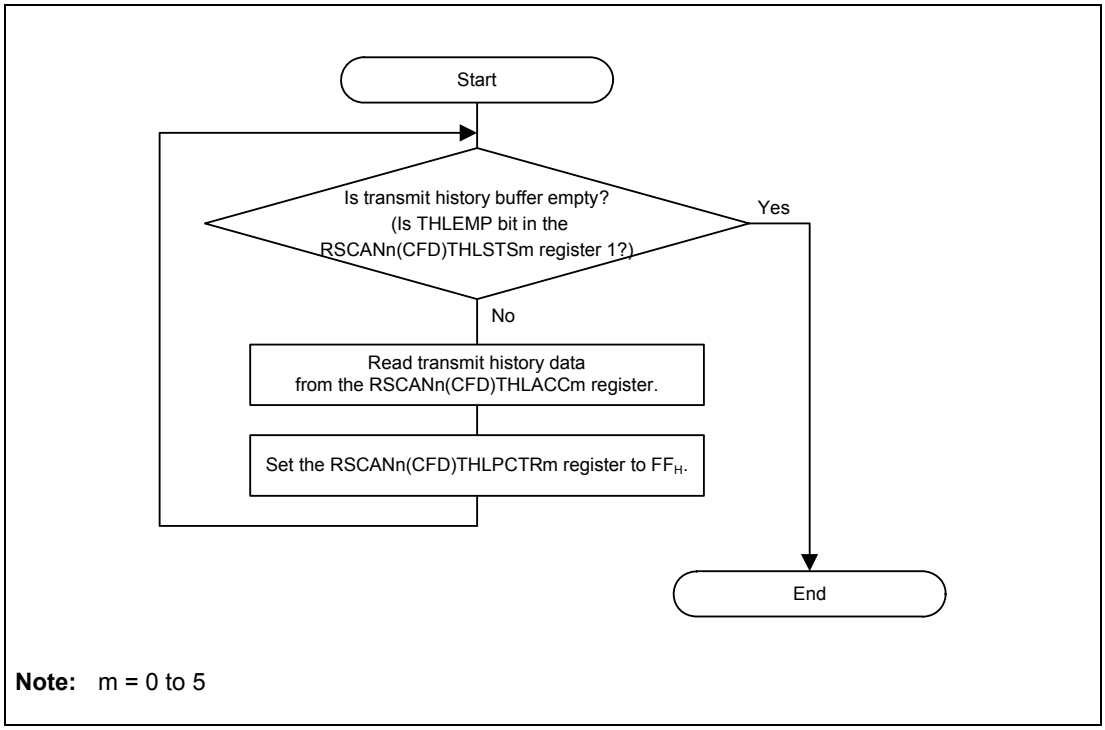



**Figure 21.33 Procedure for Transmission from the Transmit Queue**

### 21.11.3.4 Transmit History Buffer Reading Procedure

Transmit history data can be read from the RSCANn(CFD)THLACCm register. The next data can be accessed by writing $FF_H$ to the corresponding RSCANn(CFD)THLPCTRm register (m = 0 to 5) after reading a set of data. **Figure 21.34** shows the transmit history buffer reading procedure.




**Note:** m = 0 to 5

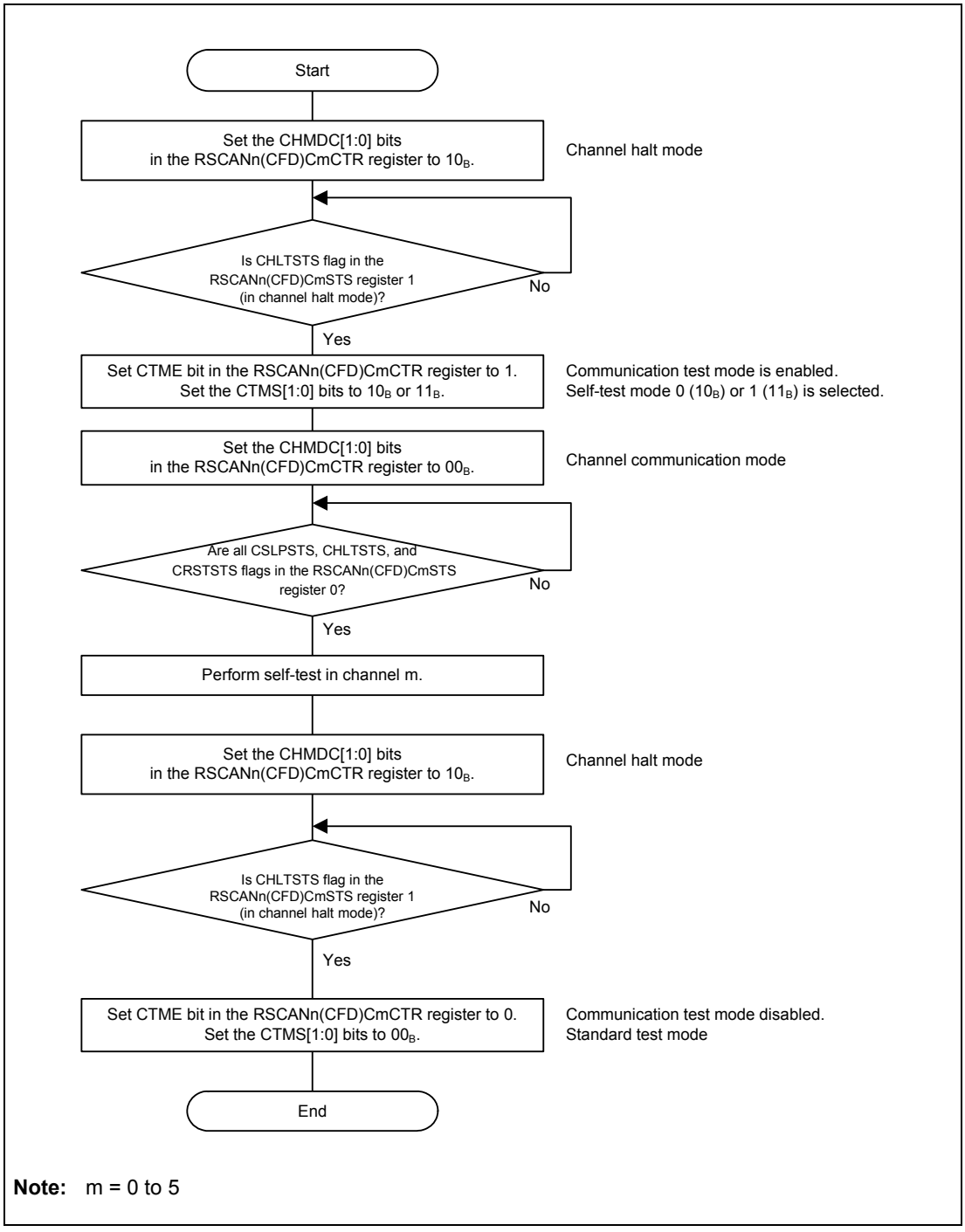
**Figure 21.34 Transmit History Buffer Reading Procedure**

## 21.11.4 Test Settings

### 21.11.4.1 Self-Test Mode Setting Procedure

Self-test mode allows communication test on a channel basis by enabling a CAN node to receive its own transmitted messages.

**Figure 21.35** shows the self-test mode setting procedure.




**Note:** m = 0 to 5

**Figure 21.35 Self-Test Mode Setting Procedure**

### 21.11.4.2 Procedure for Releasing the Protection

Since the global test function in **Table 21.192** is protected, write the protection release data 1 and release data 2 in succession to the LOCK[15:0] bits in the RSCANn(CFD)GLOCKK register, then set the target test function bit to 1.

**Table 21.192 Protection Release Data for Test Function**

| Test Function | Protection Release Data 1 | Protection Release Data 2 | Target Bit |
|---|---|---|---|
| RAM test | $7575_H$ | $8A8A_H$ | RTME bit in the RSCANn(CFD)GTSTCTR register |

If an incorrect value is written to the LOCK[15:0] bits, restart from writing the protection release data 1. **Figure 21.36** shows the procedure for releasing the protection.

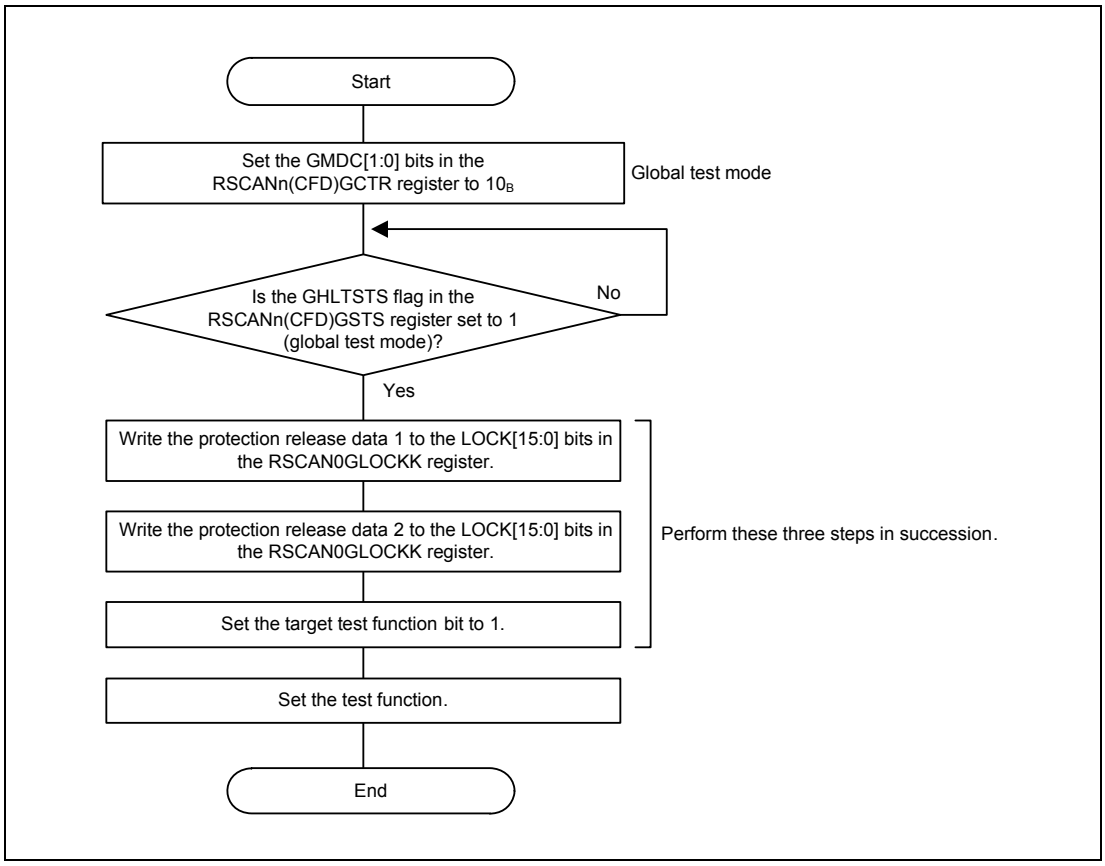



**Figure 21.36 Protection Release Procedure**

### 21.11.4.3 RAM Test Setting Procedure

RAM tests include CAN RAM read/write test. The read/write test verifies that data written to the RAM is read correctly. Before closing the RAM test, write 0000 $0000_H$ to all pages of the CAN RAM.

**Figure 21.37** shows the RAM test setting procedure.

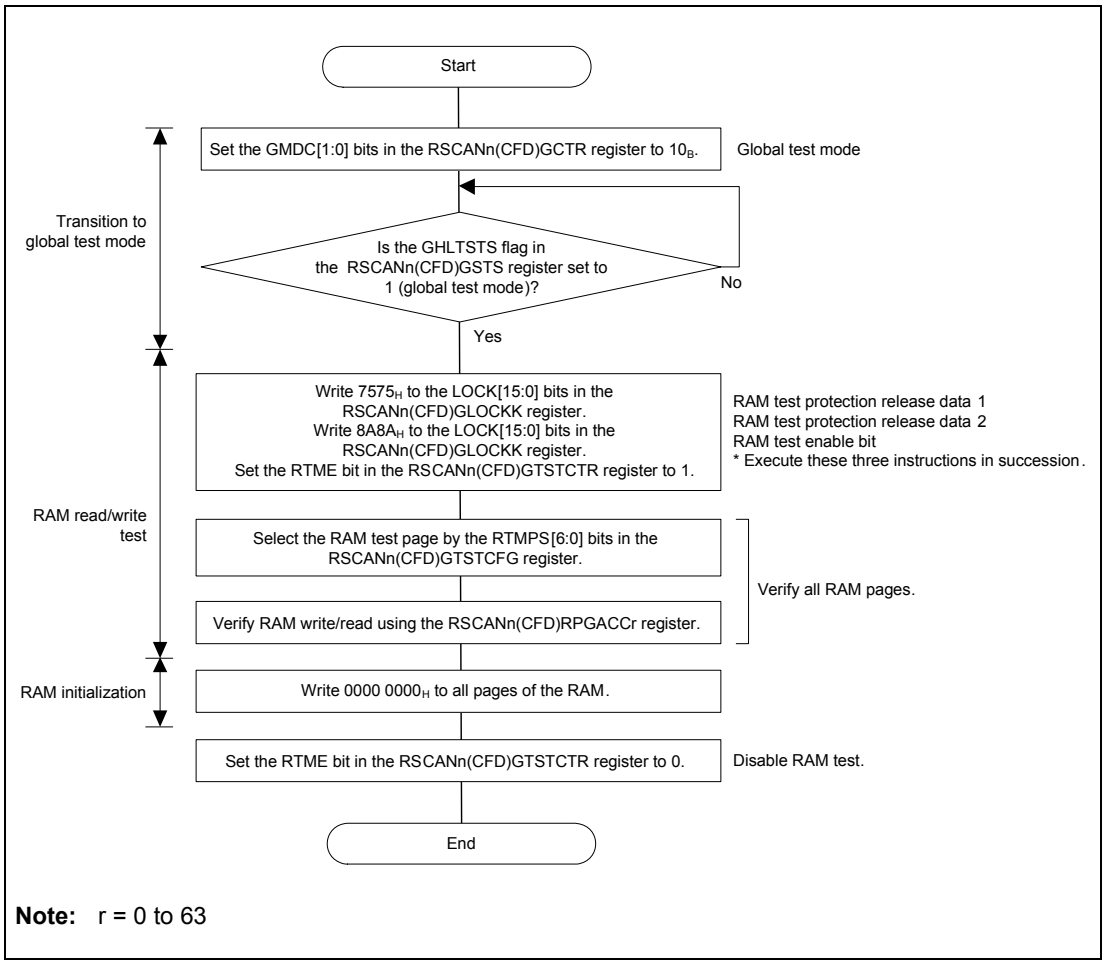
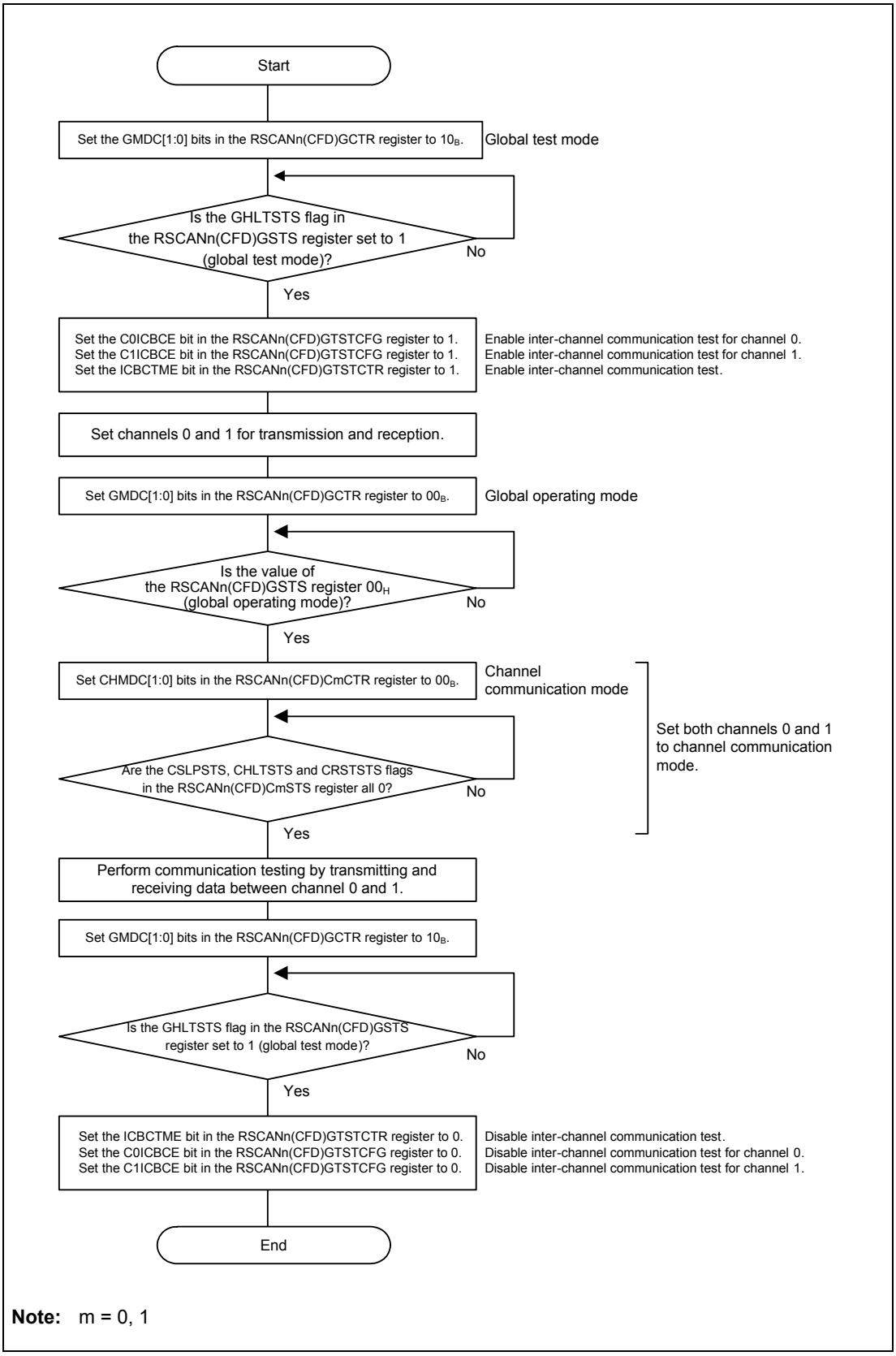



**Figure 21.37 RAM Test Setting Procedure**

#### 21.11.4.4 Inter-Channel Communication Test Setting Procedure

Communication testing can be performed by transmitting and receiving data between different channels.

**Figure 21.38** shows the inter-channel communication test setting procedure.




**Note:** m = 0, 1

**Figure 21.38 Inter-Channel Communication Test Setting Procedure (Example of Communication Test between Channel 0 and Channel 1)**

# 21.12 Detection and Correction of Errors in RS-CAN0 RAM

## 21.12.1 ECC for the RSCAN0 RAM

**Table 21.193** gives an outline of the ECC functions for the RSCAN0 RAM.

**Table 21.193 List of the ECC Functions for the RSCAN0 RAM**

| Item | Outline of Functions |
|---|---|
| ECC error detection/correction | The RAM is checked for ECC errors. The following options can be selected.<br>• 2-bit error detection and 1-bit error detection/correction<br>• 2-bit error detection and 1-bit error detection<br>The ECC error detection/correction can be disabled by using through mode. With the initial settings, error detection/correction is enabled. |
| Error notification | When an ECC error has occurred, the error is notified.<br>• Error notification can be enabled or disabled when an ECC 2-bit error is detected.<br>• Error notification can be enabled or disabled when an ECC 1-bit error is detected.<br>In the initial setting, 2-bit error notification is enabled and 1-bit error notification is disabled. However, when the interrupt is masked by the FEINTFMSK register, interrupt processing is not performed. |
| Error status | Monitoring for the detection of two-bit ECC errors and for the detection of one-bit ECC errors is available.<br>A register for clearing the error status is provided. |
| Address capture | Only one address at which an ECC error has occurred can be captured.<br>A signal is generated upon detection of ECC 2-bit or 1-bit error , and the signal is used as a trigger to capture the error-causing address (when the first (1-bit or 2-bit) error is detected after the flag is cleared). |

**CAUTION**

**When ECC error detection/correction is performed, confirm initialization of the RSCAN0 RAM (RSCAN0(CFD)GSTS.GRAMINIT = 0) before it is used.**

## 21.12.2 Interrupt Request

**Table 21.194** lists the ECC interrupt request of RSCANn RAM.

**Table 21.194 RS-CAN0 ECC Interrupt Request (FE-Level Maskable Interrupt)**

| Unit Interrupt Bit Name | Outline | Name | DMA Trigger Number |
|---|---|---|---|
| ECCDCNRAM0FEIF | RS-CAN0 ECC error interrupt | INTECCDCNRAM0 | — |

## 21.12.3 List of ECC Registers

Table 21.195 ECC Registers

| Module | Register | Symbol | Address | Guard Group |
|---|---|---|---|---|
| RS-CAN0 ECC Register (for MB RAM) | | | | |
| ECCCAN00 | ECC Control Register | ECCRCAN0CTL_PHY1 | <ECCCAN0_base > + $00_H$ | ECCCAN0 PHY1 |
| | ECC Test Mode Control Register | ECCRCAN0TMC_PHY1 | <ECCCAN0_base > + $04_H$ | ECCCAN0 PHY1 |
| | ECC Encode/Decode Input/Output Replacement Test Register | ECCRCAN0TED_PHY1 | <ECCCAN0_base > + $0C_H$ | ECCCAN0 PHY1 |
| | ECC Redundant Bit Data Control Test Register | ECCRCAN0TRC_PHY1 | <ECCCAN0_base > + $08_H$ | ECCCAN0 PHY1 |
| | ECC Decode Syndrome Data Register in ECCRCAN0TRC | ECCRCAN0SYND_PHY1 | <ECCCAN0_base > + $0B_H$ | ECCCAN0 PHY1 |
| | ECC 7-Bit Redundant Bit Data Hold Test Register in ECCRCAN0TRC | ECCRCAN0HORD_PHY1 | <ECCCAN0_base > + $0A_H$ | ECCCAN0 PHY1 |
| | ECC Encode Test Register in ECCRCAN0TRC | ECCRCAN0ECRD_PHY1 | <ECCCAN0_base > + $09_H$ | ECCCAN0 PHY1 |
| | ECC Redundant Bit Input/Output Replacement Register in ECCRCAN0TRC | ECCRCAN0ERDB_PHY1 | <ECCCAN0_base > + $08_H$ | ECCCAN0 PHY1 |
| | ECC Error Address Register 0 | ECCRCAN0AD0_PHY1 | <ECCCAN0_base> + $10_H$ | ECCCAN0 PHY1 |
| RS-CAN0 ECC Register (for AFL RAM) | | | | |
| ECCCAN01 | ECC Control Register | ECCRCANFD0CTL_PHY2 | <ECCCANFD0_base > + $00_H$ | ECCCAN0 PHY2 |
| | ECC Test Mode Control Register | ECCRCANFD0TMC_PHY2 | <ECCCANFD0_base > + $04_H$ | ECCCAN0 PHY2 |
| | ECC Encode/Decode Input/Output Replacement Test Register | ECCRCANFD0TED_PHY2 | <ECCCANFD0_base > + $0C_H$ | ECCCAN0 PHY2 |
| | ECC Redundant Bit Data Control Test Register | ECCRCANFD0TRC_PHY2 | <ECCCANFD0_base > + $08_H$ | ECCCAN0 PHY2 |
| | ECC Decode Syndrome Data Register in ECCRCANFD0TRC | ECCRCANFD0SYND_PHY2 | <ECCCANFD0_base > + $0B_H$ | ECCCAN0 PHY2 |
| | ECC 7-Bit Redundant Bit Data Hold Test Register in ECCRCANFD0TRC | ECCRCANFD0HORD_PHY2 | <ECCCANFD0_base > + $0A_H$ | ECCCAN0 PHY2 |
| | ECC Encode Test Register in ECCRCANFD0TRC | ECCRCANFD0ECRD_PHY2 | <ECCCANFD0_base > + $09_H$ | ECCCAN0 PHY2 |
| | ECC Redundant Bit Input/Output Replacement Register in ECCRCANFD0TRC | ECCRCANFD0ERDB_PHY2 | <ECCCANFD0_base > + $08_H$ | ECCCAN0 PHY2 |
| | ECC Error Address Register 0 | ECCRCANFD0AD0_PHY2 | <ECCCANFD0_base> + $10_H$ | ECCCAN0 PHY2 |

**Note:** For details about Guard Group, see **Section 33, Functional Safety**.

## 21.12.4 ECCRCAN0CTL_PHY1/ECCRCANFD0CTL_PHY2 — RSCANn ECC Control Register

The ECCRCAN0CTL_PHY1/ECCRCANFD0CTL_PHY2 register controls the mode of the ECC and the status for RSCAN0.

Bits 7, 5 and 4 should be set (written) while the RSCAN0 operation is stopped. In addition, when writing to bit 7, EMCA1 and EMCA0 need to be $01_B$.

**Access:** This register can be read or written in 16-bit units.

**Address:** <ECCCAN0_base > + $00_H$
<ECCCANFD0_base > + $00_H$

**Value after reset:** Undefined

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EMCA1 | EMCA0 | — | — | ECCOVFF | ECER2C | ECER1C | — | ECTHM | — | EC1ECP | EC2EDIC | EC1EDIC | ECER2F | ECER1F | ECEMF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | — |
| R/W | R/W*1 | R/W*1 | R | R | R | R/W*1 | R/W*1 | R | R/W | R | R/W | R/W | R/W | R | R | R |

Note 1. These bits are always read as 0.

**Table 21.196 ECCRCAN0CTL_PHY1/ECCRCANFD0CTL_PHY2 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | EMCA1 | Access control bits 1 and 0 to ECC mode selection bit<br>These bits specify whether modifying the ECTHM bit (bit 7) is disabled or enabled. The value written to these bits is not retained. When these bits are read, the read value is always 0.<br>When these bits are $01_B$, writing to bit 7 is enabled. |
| 14 | EMCA0 | |
| 13, 12 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 11 | ECCOVFF | By detecting an error while the error status is set and the new error has an address different from the one that already latched (not cleared or reset is not issued), this bit is set and error notification is generated. |

| ECCOVFF | Operation description |
|---|---|
| 0 | Overflow has not occurred after reset of clearing ECER2F and ECER1F.<br>Clearing at<br>(1) Reset<br>(2) Writing ECER2C = 1 or ECER1C = 1<br>(3) Selecting through mode enable (ECTHM = 1) |
| 1 | Error address register overflowed. |

| | | |
|---|---|---|
| 10 | ECER2C | 2-bit ECC error detection flag clear bit<br>This bit is used to clear the 2 bit error detection flag of ECER2F (bit 2).<br>This bit is always read as 0. Writing 0 is ignored. Write 1 to this bit while the ECER2F bit is set to clear the ECER2F bit. When a conflict between this bit writing and ECER2F bit setting occurs, writing to this bit has a priority. |
| 9 | ECER1C | 1-bit ECC error correction accumulation flag clear bit<br>This bit is used to clear the 1 bit error detection/correction flag of ECER1F (bit 1). This bit is always read as 0. Writing 0 is ignored. Write 1 to this bit while the ECER1F bit is set to clear the ECER1F bit. When a conflict between this bit writing and ECER1F bit setting occurs, writing to this bit has a priority. |
| 8 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

Table 21.196 ECCRCAN0CTL_PHY1/ECCRCANFD0CTL_PHY2 Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | ECTHM | ECC function through mode selection bit<br>This bit is used to set enabling and disabling of ECC.<br>Setting this bit to 1 disables ECC function.<br>When writing to this bit, (0, 1) must be written to (EMCA1, EMCA0) at the same time.<br>0: Through mode is disabled (normal operation mode).<br>1: Through mode is enabled. (ECC function disable) |
| 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 | EC1ECP | 1-bit error correction enable bit<br>This bit specifies whether to enable or disable 1-bit error correction when the ECC error detection/correction is enabled.<br>0: When 1-bit error is detected, the error will be corrected.<br>1: When 1-bit error is detected, the error will not be corrected. |
| 4 | EC2EDIC | 2-bit error detection interrupt control bit<br>This bit controls whether to generate an interrupt when 2-bit error is detected.<br>0: When 2-bit error is detected, a INTECCDCNRAMn interrupt will not be generated.<br>1: When 2-bit error is detected, a INTECCDCNRAMn interrupt will be generated. |
| 3 | EC1EDIC | 1-bit error detection interrupt control bit<br>This bit controls whether to generate an interrupt when 1-bit error is detected.<br>0: When 1-bit error is detected, a INTECCDCNRAMn interrupt will not be generated.<br>1: When 1-bit error is detected, a INTECCDCNRAMn interrupt will be generated. |
| 2 | ECER2F | 2-bit error detection flag bit<br>This flag indicates whether 2-bit error is detected during read access to the RAM when error determination is enabled (ECTHM = 0). When 2-bit error interrupt is enabled (EC2EDIC = 1) and this flag is set, an ECC 2-bit error interrupt (INTECCDCNRAMn) is output. Write 1 to the ECER2C bit (bit 10) to clear the flag. This bit is cleared at the time of through mode enable selection (ECTHM = 1). If 2-bit error is detected again while this bit is set, an interrupt signal will not be generated.<br>0: 2-bit error has not occurred since this bit was cleared.<br>1: 2-bit error has occurred. |
| 1 | ECER1F | 1-bit error detection/correction flag bit<br>This flag indicates whether 1-bit error is detected during read access to the RAM when error determination is enabled (ECTHM = 0). Write 1 to the ECER1C bit (bit 9) to clear the flag. This bit is cleared at the time of through mode enable selection (ECTHM = 1).<br>0: 1-bit error has not occurred since this bit was cleared.<br>1: 1-bit error has occurred. |
| 0 | ECEMF | ECC error message flag<br>This flag indicates whether an error exists in the current read data. This bit is updated whenever the RAM reads data. This bit might be set if it is read before initialization of the RAM. This bit is cleared at the time of through mode enable selection (ECTHM = 1) and when there is no 1 bit error in the decode circuit input data.<br>0: The currently-read RAM data does not have bit errors.<br>1: The currently-read RAM data have bit errors. |

**CAUTION**

**Bits 2 and 1 should be cleared when the ECC error message flag (ECEMF) is not set.**

**We recommend initializing the RAM before clearing bits 2 and 1.**

## 21.12.5 ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2 — RSCANn ECC Test Mode Control Register

The ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2 register switches to and controls the test mode.

This register can be used when RS-CAN is not accessed to RAM.

**Access:** This register can be read or written in 16-bit units.

**Address:** <ECCCAN0_base > + $04_H$
<ECCCANFD0_base > + $04_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ETMA1 | ETMA0 | — | — | — | — | — | — | ECTMCE | — | — | ECTRRS | ECREOS | ECENS | ECDCS | ECREIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W*1 | R/W*1 | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W |

Note 1. These bits are always read as 0.

**Table 21.197 ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | ETMA1 | Access control bits 1 and 0 to ECC test mode bit<br>These two bits specify whether updating the ECTMCE bit (bit 7) is disabled or enabled. The value written to these bits is not retained. When these bits are read, the read value is always 0.<br>When these bits are $10_B$, writing to bit 7 is enabled. |
| 14 | ETMA0 | |
| 13 to 8 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 7 | ECTMCE | ECC test mode enable bit<br>This bit specifies whether to enable access to the test registers and test control bits. When writing to this bit, (1, 0) should be written to (ETMA1, ETMA0) at the same time.<br>0: Access to the test mode registers and bits is disabled.<br>1: Access to the test mode registers and bits is enabled.<br>Test registers: ECCRCAN0TED_PHY1/ECCRCANFD0TED_PHY2, ECCRCAN0TRC_PHY1/ECCRCANFD0TRC_PHY2, ECCRCAN0SYND_PHY1/ECCRCANFD0SYND_PHY2, ECCRCAN0HORD_PHY1/ECCRCANFD0HORD_PHY2, ECCRCAN0ECRD_PHY1/ECCRCANFD0ECRD_PHY2, ECCRCAN0ERDB_PHY1/ECCRCANFD0ERDB_PHY2<br>Test control bits: ECTRRS, ECREOS, ECENS, ECDCS, ECREIS |
| 6, 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 | ECTRRS | ECC RAM read test mode selection bit<br>Select reading destination when reading ECCRCAN0TED_PHY1/ ECCRCANFD0TED_PHY2 register and reading destination when reading ECCRCAN0ERDB_PHY1/ECCRCANFD0ERDB_PHY2 register.<br>Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously). This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0:The read value of the ECCRCAN0TED_PHY1/ECCRCANFD0TED_PHY2 register is the write value of the ECCRCAN0TED_PHY1/ ECCRCANFD0TED_PHY2 register. The read value of the ECCRCAN0ERDB_PHY1/ECCRCANFD0ERDB_PHY2 register is the write value of the ECCRCAN0ERDB_PHY1/ECCRCANFD0ERDB_PHY2 register.<br>1: The read value of the ECCRCAN0TED_PHY1/ECCRCANFD0TED_PHY2 register can read RAM data. The read value of the ECCRCAN0ERDB_PHY1/ECCRCANFD0ERDB_PHY2 register is the ECC data to be written to RAM. |

**Table 21.197 ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 3 | ECREOS | ECC redundant bit output data selection bit<br>This bit is used to identify and select the ECC data to be stored in RAM as ECC data generated for write data or the value of the ECCRCAN0ERDB_PHY1/ ECCRCANFD0ERDB_PHY2 register.<br>Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously).<br>This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: Store ECC data generated for write data to RAM.<br>1: Store the value of ECCRCAN0ERDB_PHY1/ECCRCANFD0ERDB_PHY2 register to RAM. |
| 2 | ECENS | ECC encoder input selection bit<br>This bit is used to identify the targeted data for ECC data generation as write data to RAM or the value of the ECCRCAN0TED_PHY1/ ECCRCANFD0TED_PHY2 register.<br>Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously).<br>This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: Generate ECC data from the write data to RAM.<br>1: Generate ECC data from the value of the ECCRCAN0TED_PHY1/ ECCRCANFD0TED_PHY2 register. |
| 1 | ECDCS | ECC decoder input selection bit<br>This bit is used to indetifyand select the targeted data for syndrome code generation and error detection as RAM data or the value of the ECCRCAN0TED_PHY1/ECCRCANFD0TED_PHY2 register.<br>Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously). This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: Generate syndrome code from RAM data and detect errors.<br>1: Generate syndrome code from the value of the ECCRCAN0TED_PHY1/ ECCRCANFD0TED_PHY2 register and detect errors. |
| 0 | ECREIS | ECC redundant bit input data selection bit<br>This bit is used to indetifyand select the targeted data for sy nd rom code generation and error detection as RAM data or the value of the ECCRCAN0ERDB_PHY1/ECCRCANFD0ERDB_PHY2 register.<br>Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously). This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: Generate syndrome code from ECC data stored in RAM and detect errors.<br>1: Generate syndrome code from the value of the ECCRCAN0ERDB_PHY1/ ECCRCANFD0ERDB_PHY2 register and detect errors. |

## 21.12.6 ECCRCAN0TED_PHY1/ECCRCANFD0TED_PHY2 — RSCAN0 ECC Encode/ Decode Input/Output Replacement Test Register

In ECC test mode, this register handles test data.

The value of the register can be used to generate ECC data or syndrome code.

When ECC test mode is enabled (ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2.ECTMCE = 1), it is accesible. When ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2.ECTMCE = 0, writing to this register is ignored and 0000 $0000_H$ is read. This register can be used when RS-CAN is not accessed to RAM.

**Access:** This register can be read or written in 32-bit units.

**Address:** <ECCCAN0_base > + $0C_H$
<ECCCANFD0_base > + $0C_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECEDB[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECEDB[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.198 ECCRCAN0TED_PHY1/ECCRCANFD0TED_PHY2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECEDB[31:0] | When ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2.ECENS = 1, the value of this register is used to generate ECC data and the value this register to RAM is stored to RAM.<br>When ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2.ECDCS = 1, the value of this register is used to generate syndrome code and the value of this register is stored in ECC decode syndrome data register (ECCRCAN0SYND_PHY1/ECCRCANFD0SYND_PHY2).<br>In addition, when ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2. ECTRRS = 1, RAM data [31:0], instead of written data, is read for the value of this register. |

## 21.12.7 ECCRCAN0TRC_PHY1/ECCRCANFD0TRC_PHY2 — RSCAN0 ECC Redundant Bit Data Control Test Register

In ECC test mode, this test register, for ECC data, consists of four 8-bit registers, ECCRCAN0SYND_PHY1/ECCRCANFD0SYND_PHY2, ECCRCAN0HORD_PHY1/ ECCRCANFD0HORD_PHY2, ECCRCAN0ECRD_PHY1/ECCRCANFD0ECRD_PHY2, and ECCRCAN0ERDB_PHY1/ECCRCANFD0ERDB_PHY2.

When ECC test mode is enabled (ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2.ECTMCE = 1), this register can be accessed. When ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2. ECTMCE = 0, writing to this register is ignored and 0000 $0000_H$ is read. This register can be used when RS-CAN is not accessed to RAM.

**Access:** This register can be read or written in 32-bit units.

**Address:** <ECCCAN0_base > + $08_H$
<ECCCANFD0_base > + $08_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECCRCAN0SYND_PHY1/ECCRCANFD0SYND_PHY2 (See **Section 21.12.9**) | | | | | | | | ECCRCAN0HORD_PHY1/ECCRCANFD0HORD_PHY2 (See **Section 21.12.10**) | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECCRCAN0ECRD_PHY1/ECCRCANFD0ECRD_PHY2 (See **Section 21.12.11**) | | | | | | | | ECCRCAN0ERDB_PHY1/ECCRCANFD0ERDB_PHY2 (See **Section 21.12.12**) | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

## 21.12.8 ECCRCAN0AD0_PHY1/ECCRCANFD0AD0_PHY2 — RSCAN0 ECC Error Address Register 0

This is a read-only register to hold the ECC error occurring address.

When ECC error is detected while ECC error judgment is enabled, RAM address is captured by the detected signal as trigger and it is hold as the error occurring address.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <ECCCAN0_base > + $10_H$
<ECCCANFD0_base > + $10_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | ECEAD[30:16] | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECEAD[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.199 ECCRCAN0AD0_PHY1/ECCRCANFD0AD0_PHY2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | When read, the value after reset is returned. |
| 30 to 0 | ECEAD[30:0] | ECEAD0 is a read-only register to hold the address at which an ECC error has occurred.<br>If an ECC error is detected while ECC error detection is enabled, the RAM address is latched using the detection signal as a trigger, and the address is stored in ECEAD0 as the address at which the ECC error has occurred.<br>The address is stored upon detection of the first ECC error while no error status is set. However, if a 1-bit error is followed by a 2-bit error, the address of the latter is stored.<br>Only one address can be held in ECEAD0 |

## 21.12.9 ECCRCAN0SYND_PHY1/ECCRCANFD0SYND_PHY2 — RSCAN0 ECC Decode Syndrome Data Register

In ECC test mode, this is a read-only register for storing generated syndrome code.

Writing to this register is ignored.

When ECC test mode is enabled (ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2.ECTMCE = 1), this register can be accessed. When ECC test mode is disabled (ECCRCAN0TMC_PHY1/ ECCRCANFD0TMC_PHY2.ECTMCE = 0), $00_H$ is read.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <ECCCAN0_base > + $0B_H$
<ECCCANFD0_base > + $0B_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | SYND6 | SYND5 | SYND4 | SYND3 | SYND2 | SYND1 | SYND0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 21.200 ECCRCAN0SYND_PHY1/ECCRCANFD0SYND_PHY2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. |
| 6 to 0 | SYND[6:0] | The generated syndrome code is stored as needed. |

## 21.12.10 ECCRCAN0HORD_PHY1/ECCRCANFD0HORD_PHY2 — RSCAN0 ECC 7-Bit Redundant Bit Data Hold Test Register

In ECC test mode, this register is used to store ECC data for read RAM data.

Writing to this register is ignored.

When ECC test mode is enabled (ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2.ECTMCE = 1), this register can be accessed. When ECC test mode is disabled (ECCRCAN0TMC_PHY1/ ECCRCANFD0TMC_PHY2.ECTMCE = 0), $00_H$ is read.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <ECCCAN0_base > + $0A_H$
<ECCCANFD0_base > + $0A_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | HORD6 | HORD5 | HORD4 | HORD3 | HORD2 | HORD1 | HORD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 21.201 ECCRCAN0HORD_PHY1/ECCRCANFD0HORD_PHY2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. |
| 6 to 0 | HORD[6:0] | ECC code for read RAM data is stored as needed.<br>When ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2.ECTRRS = 1 and ECCRCAN0TED_PHY1/ECCRCANFD0TED_PHY2 register is read, ECC code is stored. |

## 21.12.11 ECCRCAN0ECRD_PHY1/ECCRCANFD0ECRD_PHY2 — RSCAN0 ECC Encode Test Register

In ECC test mode, this is a read-only register for storing ECC data generated for write RAM data.

Writing to this register is ignored.

When ECC test mode is enabled (ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2.ECTMCE = 1), this register is accessible. When ECC test mode is disabled (ECCRCAN0TMC_PHY1/ ECCRCANFD0TMC_PHY2.ECTMCE = 0), $00_H$ is read.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <ECCCAN0_base > + $09_H$
<ECCCANFD0_base > + $09_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | ECRD6 | ECRD5 | ECRD4 | ECRD3 | ECRD2 | ECRD1 | ECRD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 21.202 ECCRCAN0ECRD_PHY1/ECCRCANFD0ECRD_PHY2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. |
| 6 to 0 | ECRD[6:0] | These bits can read ECC data generated at the time of RAM data writing, and can read ECC data for the data written to the ECCRCAN0TED_PHY1/ ECCRCANFD0TED_PHY2 register when ECCRCAN0TMC_PHY1/ ECCRCANFD0TMC_PHY2.ECENS = 1. |

## 21.12.12 ECCRCAN0ERDB_PHY1/ECCRCANFD0ERDB_PHY2 — RSCAN0 ECC Redundant Bit Input/Output Replacement Register

In ECC test mode, this register handles ECC data.

The value of this register can be used as ECC data generated at the time of RAM writing or ECC data read at the time of RAM data reading.

When ECC test mode is enabled (ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2.ECTMCE = 1), this register is accessible. When ECC test mode is disabled (ECCRCAN0TMC_PHY1/ ECCRCANFD0TMC_PHY2.ECTMCE = 0), $00_H$ is read.

**Access:** This register can be read or written in 8-bit units.

**Address:** <ECCCAN0_base > + $08_H$
<ECCCANFD0_base > + $08_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | ERDB6 | ERDB5 | ERDB4 | ERDB3 | ERDB2 | ERDB1 | ERDB0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.203 ECCRCAN0ERDB_PHY1/ECCRCANFD0ERDB_PHY2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 6 to 0 | ERDB[6:0] | When ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2.ECREOS = 1, these bits store the value of this register as ECC data to RAM.<br>When ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2.ECREIS = 1, the value of this register is read as ECC data read from RAM.<br>When ECCRCAN0TMC_PHY1/ECCRCANFD0TMC_PHY2.ECTRRS = 1, ECC data to be stored in RAM is read instead of the write data for the read value of this register. |

## 21.12.13 SELB_READTEST — ECCREAD Test Select Register

SELB_READTEST is used to check read/write access to the CSIHn ECC registers and the RSCAN0 and the RSCAN1 ECC registers. For details, see **Section 17.7.14, SELB_READTEST — ECCREAD Test Select Register**.

## 21.13 Notes on the RS-CANFD Module

- When changing interface mode without resetting the RS-CANFD, write the value after reset to all registers and bits that are not allocated to the register map after change and then modify the RSCFDnCFDGRMCFG register.
- When changing a global mode, check the GSLPSTS, GHLTSTS, and GRSTSTS flags in the RSCANn(CFD)GSTS register for transitions. When changing a channel mode, check the CSLPSTS, CHLTSTS, and CRSTSTS flags in the RSCANn(CFD)CmSTS register (m = 0 to 5) for transitions.
- When only classical CAN frames are used in CAN FD mode, set the RSCFDnCFDCmDCFG register to the value equal to the RSCFDnCFDCmNCFG register setting value.
- The acceptance filter processing checks receive rules sequentially in ascending order from the minimum rule number. If the same ID, IDE bit, or RTR bit value is set for multiple receive rules, the minimum number of receive rule is used for the acceptance filter processing. If the message does not pass through the subsequent DLC filter processing, the data processing is terminated without returning to the acceptance filter processing and the message is not stored in the buffer.
- When linking transmit buffers to transmit/receive FIFO buffers or allocating transmit buffers to transmit queues, set the control register (RSCANn(CFD)TMCp) of the corresponding transmit buffer to $00_H$. The status register (RSCANn(CFD)TMSTSp) of the corresponding transmit buffer should not be used. Flags in other status registers (registers RSCANn(CFD)TMTRSTS0 to RSCANn(CFD)TMTRSTS2, RSCANn(CFD)TMTARSTS0 to RSCANn(CFD)TMTARSTS2, RSCANn(CFD)TMTCSTS0 to RSCANn(CFD)TMTCSTS2, and RSCANn(CFD)TMTASTS0 to RSCANn(CFD)TMTASTS2), which correspond to transmit buffers linked to transmit/receive FIFO buffers or allocated to transmit queues remain unchanged. Set the enable bit in the corresponding interrupt enable register (registers RSCANn(CFD)TMIEC0 to RSCANn(CFD)TMIEC2) to 0 (transmit buffer interrupt is disabled).
- When using transmit buffer merge mode (in CAN FD mode), write 00H to the control register (RSCANn(CFD)TMCp) of the transmit buffer corresponding to the transmit buffer allocated as a payload storage area. Set the enable bit of corresponding interrupt enable registers (RSCANn(CFD)TMIEC0 to RSCANn(CFD)TMIEC2) to 0 (to disable interrupts).
- Transmit buffers that are linked to transmit/receive FIFO buffers must not be allocated to transmit queues. Do not allocate a transmit buffer allocated as a payload storage area in transmit buffer merge mode (in CAN FD mode) to the transmit queue either.
- Only a single transmit/receive FIFO buffer can be linked to a transmit buffer. Do not link two or more transmit/receive FIFO buffers to transmit buffers of the same number.
- When the CANm bit time clock is selected as a timestamp counter clock source, the timestamp counter stops when the corresponding channel has transitioned to channel reset mode or channel halt mode.
- In case of an attempt to store a new received message when the receive FIFO buffer and the transmit/receive FIFO buffer are full, the new message is discarded. If you wish to store a new transmit message in the transmit/receive FIFO buffer or the transmit queue, check that the transmit/receive FIFO buffer or the transmit queue is not full.
- In the case of registers that access the RAM, the value after reset shown in **Section 21.4, Registers (Classical CAN Mode)** and **Section 21.5, Registers (CAN FD Mode)** indicate the values cleared by initialization of the CAN RAM. Values before clear are undefined. The following registers apply.

  - Receive rule (RSCANn(CFD)GAFLIDj, RSCANn(CFD)GAFLMj, RSCANn(CFD)GAFLP0j, RSCANn(CFD)GAFLP1j registers)
  - Receive buffers (RSCANn(CFD)RMIDq, RSCANn(CFD)RMPTRq, RSCFDnCFDRMFDSTSq, RSCANn(CFD)RMDFbq registers)
  - Receive FIFO buffer access registers (RSCANn(CFD)RFIDx, RSCANn(CFD)RFPTRx, RSCFDnCFDRFFDSTSx, and RSCANn(CFD)RFDFdx registers)
  - Transmit/receive FIFO buffer access registers (RSCANn(CFD)CFIDk, RSCANn(CFD)CFPTRk, RSCFDnCFDCFFDCSTSk, and RSCANn(CFD)CFDFdk registers)
  - Transmit buffers (RSCANn(CFD)TMIDp, RSCANn(CFD)TMPTRp, RSCFDnCFDTMFDCTRp, and RSCANn(CFD)TMDFbp registers)
  - Transmit history access register (RSCANn(CFD)THLACCm registers)
  - RAM test page access register (RSCANn(CFD)RPGACCr registers)

- The values of unused receive buffers (RSCANn(CFD)RMIDq, RSCANn(CFD)RMPTRq, RSCFDnCFDRMFDSTSq, and RSCANn(CFD)RMDFbq registers), receive FIFO buffer access registers (RSCANn(CFD)RFIDx, RSCANn(CFD)RFPTRx, RSCFDnCFDRFFDSTSx, and RSCANn(CFD)RFDFdx registers) and transmit/receive FIFO buffer access registers (RSCANn(CFD)CFIDk, RSCANn(CFD)CFPTRk, RSCFDnCFDCFFDCSTSk, and RSCANn(CFD)CFDFdk registers) are undefined when the RS-CANFD module transitions to global operation mode or global test mode after exiting from global reset mode.

## 21.14 RS-CAN

This section describes the functions and registers of RS-CAN.

**Table 21.204 Indices**

| Index | Description |
|---|---|
| n | Throughout this section, the individual RS-CAN units are generically indicated by the index “n” (n = 1); for example, RSCANnGCTR is the global control register of the RSCANn unit. |
| m | Throughout this section, the individual channels of RS-CAN units are generically indicated by the index “m” (m = 6); for example, RSCAN1CmSTS is the channel m status register. |
| i | The individual channels of RSCAN1 module are generically indicated by the index “i” (i = 0); for example, THIFi bit is the transmit history interrupt status flag of the global TX interrupt status register 0 (RSCAN1GTINTSTS0). |
| j | The individual registers associated with receive rule table are generically indicated by the index “j” (j = 0 to 15); for example, RSCAN1GAFLIDj is the receive rule ID register. |
| k | The individual transmit/receive FIFO buffers are generically indicated by the index “k” (k = 0 to [channel i × 3 + 2]); for example, RSCAN1CFCCk is the transmit/receive FIFO buffer configuration/control register. |
| x | The individual receive FIFO buffers in the RS-CAN units are identified by the index “x” (x = 0 to 7): for example, RSCAN1RFSTSx is the receive FIFO buffer status register. |
| q | The individual receive buffers are generically indicated by the index “q” (q = 0 to [channel i × 16 + 15]); for example, RSCAN1RMIDq is the receive buffer ID register. |
| p | The individual transmit buffers are generically indicated by the index “p” (q = 0 to [channel i × 16 + 15]; for example, RSCAN1TMCp is the transmit buffer control register. |
| r | The individual RAM tests for CAN are generically indicated by the index “r” (r = 0 to 63); for example, RSCAN1RPGACCr is the RAM test page access register. |
| y | The registers not covered above are indicated by the letter “y” (y = 0); for example, RSCAN1RMNDy is a receive buffer new data register. |

**Note:** The descriptions of functions and registers in this section are for the RS-CANs that has one channel (m = 6). When referring to information with indices, regard the index values as the ones corresponding to your target product. Also, note that, if the value of an index exceeds the range described in this section due to your target product, write the value after reset when writing to bits outside the index range.

The following table lists the values of indices for individual products.

**Table 21.205 Indices for Individual Products**

| Indices for Individual Products |
|---|
| **176 pins** |
| i = 0 |
| j = 0 to 15 |
| k = 0 to 2 |
| x = 0 to 7 |
| q = 0 to 15 |
| p = 0 to 15 |
| r = 0 to 63 |
| y = 0 |

### 21.14.1 Register Base Address

RSCAN1 base addresses are listed in the following table.

RSCAN1 register addresses are given as offsets from the base addresses.

**Table 21.206 Register Base Addresses**

| Base Address Name | Base Address |
|---|---|
| <RSCAN1_base> | FFD0 $8000_H$ |
| <ECCCAN1_base> | FFC7 $1020_H$ |

### 21.14.2 Clock Supply

The RSCAN1 clock supply is shown in the following table.

**Table 21.207 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| RSCAN1 | clk_xincan | CKSCLK_ICANOSC | Communicatiaon clock from osc clock |
| | clkc | CKSCLK_IPERI2 | Communication clock |
| | pclk | CKSCLK_ICAN | Module clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CKSCLK_ICAN | |

The operating frequency of the RSCAN1 depends on the transfer rate and the number of channels in use. Use the RSCAN1 within the frequency range shown in **Table 21.208**.

**Table 21.208 Range of Operating Frequency Depending on the Transfer Rate and the Number of Channels in Use in the RH850/F1K**

| Condition | | Range of Operating Frequency | | |
|---|---|---|---|---|
| Transfer Rate | No. of Channels in Use | pclk | clk_xincan*1, *3 | clkc*1, *2 |
| 1 Mbps | 1ch | pclk ≥ 18MHz | 8 MHz ≤ clk_xincan ≤ pclk/2 | 12.5 MHz ≤ clkc ≤ pclk/2 |
| 500 kbps | 1ch | pclk ≥ 8MHz | 4 MHz ≤ clk_xincan ≤ pclk/2 | 12.5 MHz ≤ clkc ≤ pclk/2 |
| 125 kbps | 1ch | pclk ≥ 8MHz | 4 MHz ≤ clk_xincan ≤ pclk/2 | 12.5 MHz ≤ clkc ≤ pclk/2 |

Note 1. Setting the DCS bit in RSCAN1GCFG enables to select either clk_xincan or clkc. Set clocks less than or equal to pclk/2 (up to 40 MHz).

Note 2. Select clk_xincan when pclk < 25 MHz.

Note 3. The maximum frequency of clk_xincan is 24 MHz.

**CAUTION**

**When the RS-CAN module is used in stop mode, set the MainOSC as the clock source of the RS-CAN module. For details about how to set the clock source, see Section 12.4.3.10, RS-CAN Clock Domains C_ISO_CAN and C_ISO_CANOSC.**

### 21.14.3 Interrupt Request

RSCAN1 interrupt requests are listed in the following table.

**Table 21.209 Interrupt Requests**

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **RSCAN1** | | | |
| INTRCANGERR1 | CAN global error interrupt | 319 | — |
| INTRCANGRECC1 | CAN receive FIFO interrupt | 320 | — |
| **CAN6** | | | |
| INTRCANmERR (m = 6) | CAN6 error interrupt | 321 | — |
| INTRCANmREC (m = 6) | CAN6 transmit/receive FIFO receive completion interrupt | 322 | — |
| INTRCANmTRX (m = 6) | CAN6 transmit interrupt | 323 | — |

**NOTE**

For the wake-up factors from standby mode, see **Section 14.1.2.1, Wake-Up Factors for Stand-By Modes**.

### 21.14.4 Reset Sources

RSCAN1 reset sources are listed in the following table. RSCAN1 is initialized by these reset sources.

**Table 21.210 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| RSCAN1 | All reset sources (ISORES) |

### 21.14.5 External Input/Output Signals

External input/output signals of RSCAN1 are listed below.

**Table 21.211 External Input/Output Signals**

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **CAN6** | | |
| CANmRX (m = 6) | CAN6 receive data input | CAN6RX |
| CANmTX (m = 6) | CAN6 transmit data output | CAN6TX |

# 21.15 Overview

## 21.15.1 Functional Overview

The RH850/F1K incorporates one CAN interface unit (RSCAN1) which consists of one channel (CAN6). **Table 21.212** shows the RS-CAN module specifications. **Figure 21.39** shows the RS-CAN module block diagram.

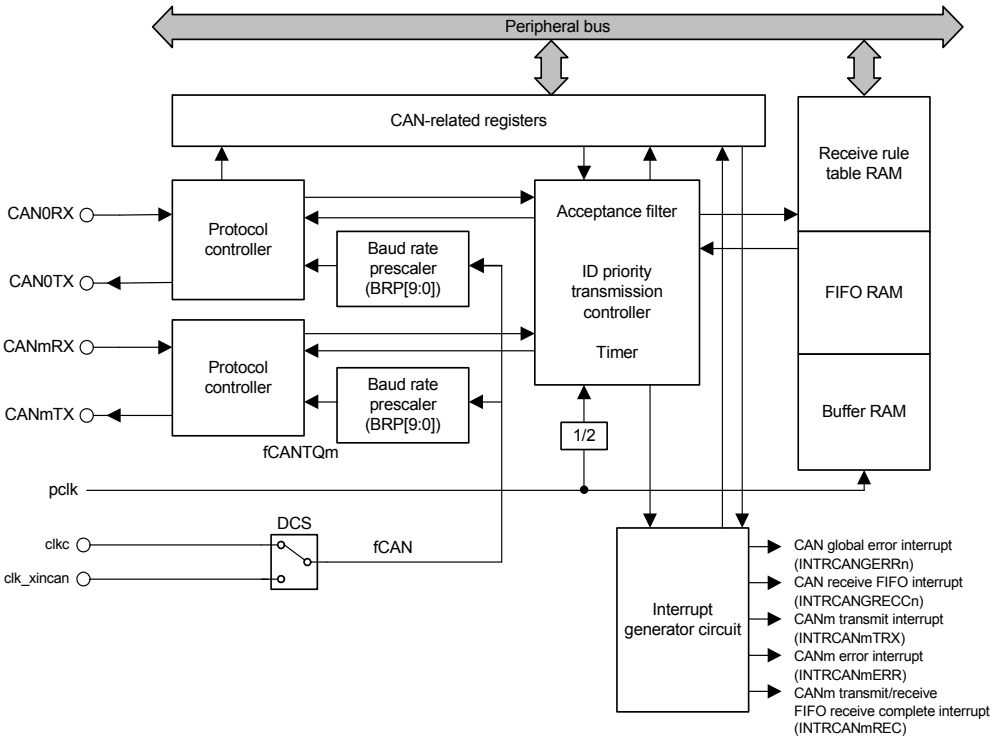
**Table 21.212 RS-CAN Module Specifications (1/2)**

| Item | Specification |
|---|---|
| Number of channels | 1 |
| Protocol | ISO11898-1 compliant |
| Communication speed | • Maximum 1 Mbps<br>Communication speed (CANm bit time clock) = $\frac{1}{\text{CANm bit time}}$<br>CANm bit time = CANmTq × Tq count per bit<br>CANmTq = $\frac{\text{(BRP[9:0] bits in the RSCAN1CmCFG register + 1)}}{\text{fCAN}}$<br>m = 6 (n = 1)<br>Tq: Time quantum<br>fCAN: Frequency of CAN clock (selected by the DCS bit in the RSCAN1GCFG register) |
| Buffer | 80 buffers (n = 1) in total<br>• Channel-dedicated: 16 buffers (16 buffers × 1 channel) (n = 1)<br>Transmit buffer: 16 buffers per channel<br>Transmit queue: Single queue per channel (shared with the transmit buffer; up to 16 buffers allocatable)<br>• Shared between channels: 64 buffers (n = 1)<br>Receive buffer: 16 buffers (n = 1)<br>Receive FIFO buffer: 8 FIFO buffers (up to 128 buffers allocatable to each)<br>Transmit/receive FIFO buffer: 3 FIFO buffers per channel (up to 128 buffers allocatable to each)<br>• ECC included |
| Reception function | • Receives data frames and remote frames.<br>• Selects ID format (standard ID, extended ID, or both IDs) to be received.<br>• Sets interrupt enable/disable for each FIFO.<br>• Mirror function (reception of messages transmitted from the own CAN node)<br>• Timestamp function (to record message reception time as a 16-bit timer value) |
| Reception filter function | • Selects receive messages according to 64 (n = 1) receive rules.<br>• Sets the number of receive rules (0 to 128) for each channel.<br>• Acceptance filter processing: Sets ID and mask for each receive rule.<br>• DLC filter processing: Enables DLC filter check for each acceptance rule. |
| Receive message transfer function | • Routing function<br>Transfers receive messages to arbitrary destinations (can be transferred to up to 8 buffers)<br>Transfer destination: Receive buffer, receive FIFO buffer, and/or transmit/receive FIFO buffer<br>• Labeling function<br>Stores label information together with a message in a receive buffer and FIFO buffer. |
| Transmission function | • Transmits data frames and remote frames.<br>• Selects ID format (standard ID, extended ID, or both IDs) to be transmitted.<br>• Sets interrupt enable/disable for each transmit buffer and transmit/receive FIFO buffer.<br>• Selects ID priority transmission or transmit buffer number priority transmission.<br>• Transmission can be aborted (possible to confirm with a flag)<br>• One-shot transmission function |
| Interval transmission function | Transmits messages at configurable intervals<br>(transmit mode of transmit/receive FIFO buffers) |

Table 21.212 RS-CAN Module Specifications (2/2)

| Item | Specification |
|---|---|
| Transmit queue function | Transmits all stored messages according to the ID priority. |
| Transmit history function | Stores the history information of transmission-completed messages |
| Gateway function | Not supported. |
| Bus off recovery mode selection | Selects the method for returning from bus off state.<br>• ISO11898-1 compliant<br>• Automatic entry to channel halt mode at bus-off entry<br>• Automatic entry to channel halt mode at bus-off end<br>• Transition to channel halt mode by program request<br>• Transition to the error-active state by program request (forcible return from the bus off state) |
| Error status monitoring | • Monitors CAN protocol errors (stuff error, form error, ACK error, CRC error, bit error, ACK delimiter error, and bus dominant lock).<br>• Detects error status transitions (error warning, error passive, bus off entry, and bus off recovery)<br>• Reads the error counter.<br>• Monitors DLC errors. |
| Interrupt source | 5 sources<br>• Global interrupts (2 sources/unit)<br>Receive FIFO interrupt (1 source/unit)<br>Global error interrupt (1 source/unit)<br>• Channel interrupts (3 sources/channel)<br>CANm transmit interrupt (m = 6)<br>– CANm transmit complete interrupt<br>– CANm transmit abort interrupt<br>– CANm transmit/receive FIFO transmit complete interrupt (in transmit mode)<br>– CANm transmit history interrupt<br>– CANm transmit queue interrupt<br>CANm transmit/receive FIFO receive complete interrupt (in receive mode)<br>CANm error interrupt |
| CAN stop mode | Reduces power consumption by stopping clock supply to the RS-CAN module. |
| CAN clock source | Selects the clkc or the clk_xincan.<br>For the operating frequency range, see **Table 21.208**. |
| Test function | Test function for user evaluation<br>• Listen-only mode<br>• Self-test mode 0 (external loopback)<br>• Self-test mode 1 (internal loopback)<br>• RAM test (read/write test) |

## 21.15.2 Block Diagram




**Note:** n = 1, m = 6
BRP[9:0]: Bits in the RSCAN1CmCFG register
DCS: Bits in the RSCAN1GCFG register
fCANTQm: CANmTq clock
fCAN: CAN clock

**Figure 21.39 RS-CAN Module Block Diagram**

# 21.16 Registers

## 21.16.1 List of Registers

RS-CAN registers are listed in the following table.

For details about <RSCAN1_base>, see **Section 21.14.1, Register Base Address**.

**Table 21.213 Registers (1/9)**

| Module | Register | Symbol | Address | Guard Group |
|---|---|---|---|---|
| RSCAN1 | Channel m configuration register | RSCAN1CmCFG | <RSCAN1_base> + $0000_H$ | RSCAN1 Ch0 |
| | Channel m control register | RSCAN1CmCTR | <RSCAN1_base> + $0004_H$ | RSCAN1 Ch0 |
| | Channel m status register | RSCAN1CmSTS | <RSCAN1_base> + $0008_H$ | RSCAN1 Ch0 |
| | Channel m error flag register | RSCAN1CmERFL | <RSCAN1_base> + $000C_H$ | RSCAN1 Ch0 |
| | Global configuration register | RSCAN1GCFG | <RSCAN1_base> + $0084_H$ | RSCAN1 Global |
| | Global control register | RSCAN1GCTR | <RSCAN1_base> + $0088_H$ | RSCAN1 Global |
| | Global status register | RSCAN1GSTS | <RSCAN1_base> + $008C_H$ | RSCAN1 Global |
| | Global error flag register | RSCAN1GERFL | <RSCAN1_base> + $0090_H$ | RSCAN1 Global |
| | Global timestamp counter register | RSCAN1GTSC | <RSCAN1_base> + $0094_H$ | RSCAN1 Global |
| | Receive rule entry control register | RSCAN1GAFLECTR | <RSCAN1_base> + $0098_H$ | RSCAN1 Global |
| | Receive rule configuration register 0 | RSCAN1GAFLCFG0 | <RSCAN1_base> + $009C_H$ | RSCAN1 Global |
| | Receive buffer number register | RSCAN1RMNB | <RSCAN1_base> + $00A4_H$ | RSCAN1 Global |
| | Receive buffer new data register 0 | RSCAN1RMND0 | <RSCAN1_base> + $00A8_H$ | RSCAN1 Global |
| | Receive FIFO buffer configuration and control register 0 | RSCAN1RFCC0 | <RSCAN1_base> + $00B8_H$ | RSCAN1 Global |
| | Receive FIFO buffer configuration and control register 1 | RSCAN1RFCC1 | <RSCAN1_base> + $00BC_H$ | RSCAN1 Global |
| | Receive FIFO buffer configuration and control register 2 | RSCAN1RFCC2 | <RSCAN1_base> + $00C0_H$ | RSCAN1 Global |
| | Receive FIFO buffer configuration and control register 3 | RSCAN1RFCC3 | <RSCAN1_base> + $00C4_H$ | RSCAN1 Global |
| | Receive FIFO buffer configuration and control register 4 | RSCAN1RFCC4 | <RSCAN1_base> + $00C8_H$ | RSCAN1 Global |
| | Receive FIFO buffer configuration and control register 5 | RSCAN1RFCC5 | <RSCAN1_base> + $00CC_H$ | RSCAN1 Global |
| | Receive FIFO buffer configuration and control register 6 | RSCAN1RFCC6 | <RSCAN1_base> + $00D0_H$ | RSCAN1 Global |
| | Receive FIFO buffer configuration and control register 7 | RSCAN1RFCC7 | <RSCAN1_base> + $00D4_H$ | RSCAN1 Global |
| | Receive FIFO buffer status register 0 | RSCAN1RFSTS0 | <RSCAN1_base> + $00D8_H$ | RSCAN1 Global |
| | Receive FIFO buffer status register 1 | RSCAN1RFSTS1 | <RSCAN1_base> + $00DC_H$ | RSCAN1 Global |
| | Receive FIFO buffer status register 2 | RSCAN1RFSTS2 | <RSCAN1_base> + $00E0_H$ | RSCAN1 Global |
| | Receive FIFO buffer status register 3 | RSCAN1RFSTS3 | <RSCAN1_base> + $00E4_H$ | RSCAN1 Global |
| | Receive FIFO buffer status register 4 | RSCAN1RFSTS4 | <RSCAN1_base> + $00E8_H$ | RSCAN1 Global |
| | Receive FIFO buffer status register 5 | RSCAN1RFSTS5 | <RSCAN1_base> + $00EC_H$ | RSCAN1 Global |
| | Receive FIFO buffer status register 6 | RSCAN1RFSTS6 | <RSCAN1_base> + $00F0_H$ | RSCAN1 Global |
| | Receive FIFO buffer status register 7 | RSCAN1RFSTS7 | <RSCAN1_base> + $00F4_H$ | RSCAN1 Global |
| | Receive FIFO buffer pointer control register 0 | RSCAN1RFPCTR0 | <RSCAN1_base> + $00F8_H$ | RSCAN1 Global |
| | Receive FIFO buffer pointer control register 1 | RSCAN1RFPCTR1 | <RSCAN1_base> + $00FC_H$ | RSCAN1 Global |
| | Receive FIFO buffer pointer control register 2 | RSCAN1RFPCTR2 | <RSCAN1_base> + $0100_H$ | RSCAN1 Global |
| | Receive FIFO buffer pointer control register 3 | RSCAN1RFPCTR3 | <RSCAN1_base> + $0104_H$ | RSCAN1 Global |
| | Receive FIFO buffer pointer control register 4 | RSCAN1RFPCTR4 | <RSCAN1_base> + $0108_H$ | RSCAN1 Global |
| | Receive FIFO buffer pointer control register 5 | RSCAN1RFPCTR5 | <RSCAN1_base> + $010C_H$ | RSCAN1 Global |
| | Receive FIFO buffer pointer control register 6 | RSCAN1RFPCTR6 | <RSCAN1_base> + $0110_H$ | RSCAN1 Global |
| | Receive FIFO buffer pointer control register 7 | RSCAN1RFPCTR7 | <RSCAN1_base> + $0114_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer configuration and control register 0 | RSCAN1CFCC0 | <RSCAN1_base> + $0118_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer configuration and control register 1 | RSCAN1CFCC1 | <RSCAN1_base> + $011C_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer configuration and control register 2 | RSCAN1CFCC2 | <RSCAN1_base> + $0120_H$ | RSCAN1 Global |

Table 21.213 Registers (2/9)

| Module | Register | Symbol | Address | Guard Group |
|---|---|---|---|---|
| RSCAN1 | Transmit/receive FIFO buffer status register 0 | RSCAN1CFSTS0 | <RSCAN1_base> + $0178_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer status register 1 | RSCAN1CFSTS1 | <RSCAN1_base> + $017C_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer status register 2 | RSCAN1CFSTS2 | <RSCAN1_base> + $0180_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer pointer control register 0 | RSCAN1CFPCTR0 | <RSCAN1_base> + $01D8_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer pointer control register 1 | RSCAN1CFPCTR1 | <RSCAN1_base> + $01DC_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer pointer control register 2 | RSCAN1CFPCTR2 | <RSCAN1_base> + $01E0_H$ | RSCAN1 Global |
| | FIFO empty status register | RSCAN1FESTS | <RSCAN1_base> + $0238_H$ | RSCAN1 Global |
| | FIFO full status register | RSCAN1FFSTS | <RSCAN1_base> + $023C_H$ | RSCAN1 Global |
| | FIFO Msg lost status register | RSCAN1FMSTS | <RSCAN1_base> + $0240_H$ | RSCAN1 Global |
| | Receive FIFO buffer interrupt flag status register | RSCAN1RFISTS | <RSCAN1_base> + $0244_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer RX interrupt flag status register | RSCAN1CFRISTS | <RSCAN1_base> + $0248_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer TX interrupt flag status register | RSCAN1CFTISTS | <RSCAN1_base> + $024C_H$ | RSCAN1 Global |
| | Transmit buffer control register 0 | RSCAN1TMC0 | <RSCAN1_base> + $0250_H$ | RSCAN1 Global |
| | Transmit buffer control register 1 | RSCAN1TMC1 | <RSCAN1_base> + $0251_H$ | RSCAN1 Global |
| | Transmit buffer control register 2 | RSCAN1TMC2 | <RSCAN1_base> + $0252_H$ | RSCAN1 Global |
| | Transmit buffer control register 3 | RSCAN1TMC3 | <RSCAN1_base> + $0253_H$ | RSCAN1 Global |
| | Transmit buffer control register 4 | RSCAN1TMC4 | <RSCAN1_base> + $0254_H$ | RSCAN1 Global |
| | Transmit buffer control register 5 | RSCAN1TMC5 | <RSCAN1_base> + $0255_H$ | RSCAN1 Global |
| | Transmit buffer control register 6 | RSCAN1TMC6 | <RSCAN1_base> + $0256_H$ | RSCAN1 Global |
| | Transmit buffer control register 7 | RSCAN1TMC7 | <RSCAN1_base> + $0257_H$ | RSCAN1 Global |
| | Transmit buffer control register 8 | RSCAN1TMC8 | <RSCAN1_base> + $0258_H$ | RSCAN1 Global |
| | Transmit buffer control register 9 | RSCAN1TMC9 | <RSCAN1_base> + $0259_H$ | RSCAN1 Global |
| | Transmit buffer control register 10 | RSCAN1TMC10 | <RSCAN1_base> + $025A_H$ | RSCAN1 Global |
| | Transmit buffer control register 11 | RSCAN1TMC11 | <RSCAN1_base> + $025B_H$ | RSCAN1 Global |
| | Transmit buffer control register 12 | RSCAN1TMC12 | <RSCAN1_base> + $025C_H$ | RSCAN1 Global |
| | Transmit buffer control register 13 | RSCAN1TMC13 | <RSCAN1_base> + $025D_H$ | RSCAN1 Global |
| | Transmit buffer control register 14 | RSCAN1TMC14 | <RSCAN1_base> + $025E_H$ | RSCAN1 Global |
| | Transmit buffer control register 15 | RSCAN1TMC15 | <RSCAN1_base> + $025F_H$ | RSCAN1 Global |
| | Transmit buffer status register 0 | RSCAN1TMSTS0 | <RSCAN1_base> + $02D0_H$ | RSCAN1 Global |
| | Transmit buffer status register 1 | RSCAN1TMSTS1 | <RSCAN1_base> + $02D1_H$ | RSCAN1 Global |
| | Transmit buffer status register 2 | RSCAN1TMSTS2 | <RSCAN1_base> + $02D2_H$ | RSCAN1 Global |
| | Transmit buffer status register 3 | RSCAN1TMSTS3 | <RSCAN1_base> + $02D3_H$ | RSCAN1 Global |
| | Transmit buffer status register 4 | RSCAN1TMSTS4 | <RSCAN1_base> + $02D4_H$ | RSCAN1 Global |
| | Transmit buffer status register 5 | RSCAN1TMSTS5 | <RSCAN1_base> + $02D5_H$ | RSCAN1 Global |
| | Transmit buffer status register 6 | RSCAN1TMSTS6 | <RSCAN1_base> + $02D6_H$ | RSCAN1 Global |
| | Transmit buffer status register 7 | RSCAN1TMSTS7 | <RSCAN1_base> + $02D7_H$ | RSCAN1 Global |
| | Transmit buffer status register 8 | RSCAN1TMSTS8 | <RSCAN1_base> + $02D8_H$ | RSCAN1 Global |
| | Transmit buffer status register 9 | RSCAN1TMSTS9 | <RSCAN1_base> + $02D9_H$ | RSCAN1 Global |
| | Transmit buffer status register 10 | RSCAN1TMSTS10 | <RSCAN1_base> + $02DA_H$ | RSCAN1 Global |
| | Transmit buffer status register 11 | RSCAN1TMSTS11 | <RSCAN1_base> + $02DB_H$ | RSCAN1 Global |
| | Transmit buffer status register 12 | RSCAN1TMSTS12 | <RSCAN1_base> + $02DC_H$ | RSCAN1 Global |
| | Transmit buffer status register 13 | RSCAN1TMSTS13 | <RSCAN1_base> + $02DD_H$ | RSCAN1 Global |
| | Transmit buffer status register 14 | RSCAN1TMSTS14 | <RSCAN1_base> + $02DE_H$ | RSCAN1 Global |
| | Transmit buffer status register 15 | RSCAN1TMSTS15 | <RSCAN1_base> + $02DF_H$ | RSCAN1 Global |
| | Transmit buffer transmit request status register 0 | RSCAN1TMTRSTS0 | <RSCAN1_base> + $0350_H$ | RSCAN1 Global |
| | Transmit buffer transmit abort request status register 0 | RSCAN1TMTARSTS0 | <RSCAN1_base> + $0360_H$ | RSCAN1 Global |
| | Transmit buffer transmit complete status register 0 | RSCAN1TMTCSTS0 | <RSCAN1_base> + $0370_H$ | RSCAN1 Global |

Table 21.213 Registers (3/9)

| Module | Register | Symbol | Address | Guard Group |
|---|---|---|---|---|
| RSCAN1 | Transmit buffer transmit abort status register 0 | RSCAN1TMTASTS0 | <RSCAN1_base> + $0380_H$ | RSCAN1 Global |
| | Transmit buffer interrupt enable configuration register 0 | RSCAN1TMIEC0 | <RSCAN1_base> + $0390_H$ | RSCAN1 Global |
| | Transmit queue configuration and control register m | RSCAN1TXQCCm | <RSCAN1_base> + $03A0_H$ | RSCAN1 Ch0 |
| | Transmit queue status register m | RSCAN1TXQSTSm | <RSCAN1_base> + $03C0_H$ | RSCAN1 Ch0 |
| | Transmit queue pointer control register m | RSCAN1TXQPCTRm | <RSCAN1_base> + $03E0_H$ | RSCAN1 Ch0 |
| | Transmit history configuration and control register m | RSCAN1THLCCm | <RSCAN1_base> + $0400_H$ | RSCAN1 Ch0 |
| | Transmit history status register m | RSCAN1THLSTSm | <RSCAN1_base> + $0420_H$ | RSCAN1 Ch0 |
| | Transmit history pointer control register m | RSCAN1THLPCTRm | <RSCAN1_base> + $0440_H$ | RSCAN1 Ch0 |
| | Global TX interrupt status register 0 | RSCAN1GTINTSTS0 | <RSCAN1_base> + $0460_H$ | RSCAN1 Global |
| | Global test configuration register | RSCAN1GTSTCFG | <RSCAN1_base> + $0468_H$ | RSCAN1 Global |
| | Global test control register | RSCAN1GTSTCTR | <RSCAN1_base> + $046C_H$ | RSCAN1 Global |
| | Global lock key register | RSCAN1GLOCKK | <RSCAN1_base> + $047C_H$ | RSCAN1 Global |
| | Receive rule ID register 0 | RSCAN1GAFLID0 | <RSCAN1_base> + $0500_H$ | RSCAN1 Global |
| | Receive rule mask register 0 | RSCAN1GAFLM0 | <RSCAN1_base> + $0504_H$ | RSCAN1 Global |
| | Receive rule pointer 0 register 0 | RSCAN1GAFLP00 | <RSCAN1_base> + $0508_H$ | RSCAN1 Global |
| | Receive rule pointer 1 register 0 | RSCAN1GAFLP10 | <RSCAN1_base> + $050C_H$ | RSCAN1 Global |
| | Receive rule ID register 1 | RSCAN1GAFLID1 | <RSCAN1_base> + $0510_H$ | RSCAN1 Global |
| | Receive rule mask register 1 | RSCAN1GAFLM1 | <RSCAN1_base> + $0514_H$ | RSCAN1 Global |
| | Receive rule pointer 0 register 1 | RSCAN1GAFLP01 | <RSCAN1_base> + $0518_H$ | RSCAN1 Global |
| | Receive rule pointer 1 register 1 | RSCAN1GAFLP11 | <RSCAN1_base> + $051C_H$ | RSCAN1 Global |
| | Receive rule ID register 2 | RSCAN1GAFLID2 | <RSCAN1_base> + $0520_H$ | RSCAN1 Global |
| | Receive rule mask register 2 | RSCAN1GAFLM2 | <RSCAN1_base> + $0524_H$ | RSCAN1 Global |
| | Receive rule pointer 0 register 2 | RSCAN1GAFLP02 | <RSCAN1_base> + $0528_H$ | RSCAN1 Global |
| | Receive rule pointer 1 register 2 | RSCAN1GAFLP12 | <RSCAN1_base> + $052C_H$ | RSCAN1 Global |
| | Receive rule ID register 3 | RSCAN1GAFLID3 | <RSCAN1_base> + $0530_H$ | RSCAN1 Global |
| | Receive rule mask register 3 | RSCAN1GAFLM3 | <RSCAN1_base> + $0534_H$ | RSCAN1 Global |
| | Receive rule pointer 0 register 3 | RSCAN1GAFLP03 | <RSCAN1_base> + $0538_H$ | RSCAN1 Global |
| | Receive rule pointer 1 register 3 | RSCAN1GAFLP13 | <RSCAN1_base> + $053C_H$ | RSCAN1 Global |
| | Receive rule ID register 4 | RSCAN1GAFLID4 | <RSCAN1_base> + $0540_H$ | RSCAN1 Global |
| | Receive rule mask register 4 | RSCAN1GAFLM4 | <RSCAN1_base> + $0544_H$ | RSCAN1 Global |
| | Receive rule pointer 0 register 4 | RSCAN1GAFLP04 | <RSCAN1_base> + $0548_H$ | RSCAN1 Global |
| | Receive rule pointer 1 register 4 | RSCAN1GAFLP14 | <RSCAN1_base> + $054C_H$ | RSCAN1 Global |
| | Receive rule ID register 5 | RSCAN1GAFLID5 | <RSCAN1_base> + $0550_H$ | RSCAN1 Global |
| | Receive rule mask register 5 | RSCAN1GAFLM5 | <RSCAN1_base> + $0554_H$ | RSCAN1 Global |
| | Receive rule pointer 0 register 5 | RSCAN1GAFLP05 | <RSCAN1_base> + $0558_H$ | RSCAN1 Global |
| | Receive rule pointer 1 register 5 | RSCAN1GAFLP15 | <RSCAN1_base> + $055C_H$ | RSCAN1 Global |
| | Receive rule ID register 6 | RSCAN1GAFLID6 | <RSCAN1_base> + $0560_H$ | RSCAN1 Global |
| | Receive rule mask register 6 | RSCAN1GAFLM6 | <RSCAN1_base> + $0564_H$ | RSCAN1 Global |
| | Receive rule pointer 0 register 6 | RSCAN1GAFLP06 | <RSCAN1_base> + $0568_H$ | RSCAN1 Global |
| | Receive rule pointer 1 register 6 | RSCAN1GAFLP16 | <RSCAN1_base> + $056C_H$ | RSCAN1 Global |
| | Receive rule ID register 7 | RSCAN1GAFLID7 | <RSCAN1_base> + $0570_H$ | RSCAN1 Global |
| | Receive rule mask register 7 | RSCAN1GAFLM7 | <RSCAN1_base> + $0574_H$ | RSCAN1 Global |
| | Receive rule pointer 0 register 7 | RSCAN1GAFLP07 | <RSCAN1_base> + $0578_H$ | RSCAN1 Global |
| | Receive rule pointer 1 register 7 | RSCAN1GAFLP17 | <RSCAN1_base> + $057C_H$ | RSCAN1 Global |
| | Receive rule ID register 8 | RSCAN1GAFLID8 | <RSCAN1_base> + $0580_H$ | RSCAN1 Global |
| | Receive rule mask register 8 | RSCAN1GAFLM8 | <RSCAN1_base> + $0584_H$ | RSCAN1 Global |
| | Receive rule pointer 0 register 8 | RSCAN1GAFLP08 | <RSCAN1_base> + $0588_H$ | RSCAN1 Global |

Table 21.213 Registers (4/9)

| Module | Register | Symbol | Address | Guard Group |
|---|---|---|---|---|
| RSCAN1 | Receive rule pointer 1 register 8 | RSCAN1GAFLP18 | <RSCAN1_base> + 058C$_H$ | RSCAN1 Global |
| | Receive rule ID register 9 | RSCAN1GAFLID9 | <RSCAN1_base> + 0590$_H$ | RSCAN1 Global |
| | Receive rule mask register 9 | RSCAN1GAFLM9 | <RSCAN1_base> + 0594$_H$ | RSCAN1 Global |
| | Receive rule pointer 0 register 9 | RSCAN1GAFLP09 | <RSCAN1_base> + 0598$_H$ | RSCAN1 Global |
| | Receive rule pointer 1 register 9 | RSCAN1GAFLP19 | <RSCAN1_base> + 059C$_H$ | RSCAN1 Global |
| | Receive rule ID register 10 | RSCAN1GAFLID10 | <RSCAN1_base> + 05A0$_H$ | RSCAN1 Global |
| | Receive rule mask register 10 | RSCAN1GAFLM10 | <RSCAN1_base> + 05A4$_H$ | RSCAN1 Global |
| | Receive rule pointer 0 register 10 | RSCAN1GAFLP010 | <RSCAN1_base> + 05A8$_H$ | RSCAN1 Global |
| | Receive rule pointer 1 register 10 | RSCAN1GAFLP110 | <RSCAN1_base> + 05AC$_H$ | RSCAN1 Global |
| | Receive rule ID register 11 | RSCAN1GAFLID11 | <RSCAN1_base> + 05B0$_H$ | RSCAN1 Global |
| | Receive rule mask register 11 | RSCAN1GAFLM11 | <RSCAN1_base> + 05B4$_H$ | RSCAN1 Global |
| | Receive rule pointer 0 register 11 | RSCAN1GAFLP011 | <RSCAN1_base> + 05B8$_H$ | RSCAN1 Global |
| | Receive rule pointer 1 register 11 | RSCAN1GAFLP111 | <RSCAN1_base> + 05BC$_H$ | RSCAN1 Global |
| | Receive rule ID register 12 | RSCAN1GAFLID12 | <RSCAN1_base> + 05C0$_H$ | RSCAN1 Global |
| | Receive rule mask register 12 | RSCAN1GAFLM12 | <RSCAN1_base> + 05C4$_H$ | RSCAN1 Global |
| | Receive rule pointer 0 register 12 | RSCAN1GAFLP012 | <RSCAN1_base> + 05C8$_H$ | RSCAN1 Global |
| | Receive rule pointer 1 register 12 | RSCAN1GAFLP112 | <RSCAN1_base> + 05CC$_H$ | RSCAN1 Global |
| | Receive rule ID register 13 | RSCAN1GAFLID13 | <RSCAN1_base> + 05D0$_H$ | RSCAN1 Global |
| | Receive rule mask register 13 | RSCAN1GAFLM13 | <RSCAN1_base> + 05D4$_H$ | RSCAN1 Global |
| | Receive rule pointer 0 register 13 | RSCAN1GAFLP013 | <RSCAN1_base> + 05D8$_H$ | RSCAN1 Global |
| | Receive rule pointer 1 register 13 | RSCAN1GAFLP113 | <RSCAN1_base> + 05DC$_H$ | RSCAN1 Global |
| | Receive rule ID register 14 | RSCAN1GAFLID14 | <RSCAN1_base> + 05E0$_H$ | RSCAN1 Global |
| | Receive rule mask register 14 | RSCAN1GAFLM14 | <RSCAN1_base> + 05E4$_H$ | RSCAN1 Global |
| | Receive rule pointer 0 register 14 | RSCAN1GAFLP014 | <RSCAN1_base> + 05E8$_H$ | RSCAN1 Global |
| | Receive rule pointer 1 register 14 | RSCAN1GAFLP114 | <RSCAN1_base> + 05EC$_H$ | RSCAN1 Global |
| | Receive rule ID register 15 | RSCAN1GAFLID15 | <RSCAN1_base> + 05F0$_H$ | RSCAN1 Global |
| | Receive rule mask register 15 | RSCAN1GAFLM15 | <RSCAN1_base> + 05F4$_H$ | RSCAN1 Global |
| | Receive rule pointer 0 register 15 | RSCAN1GAFLP015 | <RSCAN1_base> + 05F8$_H$ | RSCAN1 Global |
| | Receive rule pointer 1 register 15 | RSCAN1GAFLP115 | <RSCAN1_base> + 05FC$_H$ | RSCAN1 Global |
| | Receive buffer ID register 0 | RSCAN1RMID0 | <RSCAN1_base> + 0600$_H$ | RSCAN1 Global |
| | Receive buffer pointer register 0 | RSCAN1RMPTR0 | <RSCAN1_base> + 0604$_H$ | RSCAN1 Global |
| | Receive buffer data field 0 register 0 | RSCAN1RMDF00 | <RSCAN1_base> + 0608$_H$ | RSCAN1 Global |
| | Receive buffer data field 1 register 0 | RSCAN1RMDF10 | <RSCAN1_base> + 060C$_H$ | RSCAN1 Global |
| | Receive buffer ID register 1 | RSCAN1RMID1 | <RSCAN1_base> + 0610$_H$ | RSCAN1 Global |
| | Receive buffer pointer register 1 | RSCAN1RMPTR1 | <RSCAN1_base> + 0614$_H$ | RSCAN1 Global |
| | Receive buffer data field 0 register 1 | RSCAN1RMDF01 | <RSCAN1_base> + 0618$_H$ | RSCAN1 Global |
| | Receive buffer data field 1 register 1 | RSCAN1RMDF11 | <RSCAN1_base> + 061C$_H$ | RSCAN1 Global |
| | Receive buffer ID register 2 | RSCAN1RMID2 | <RSCAN1_base> + 0620$_H$ | RSCAN1 Global |
| | Receive buffer pointer register 2 | RSCAN1RMPTR2 | <RSCAN1_base> + 0624$_H$ | RSCAN1 Global |
| | Receive buffer data field 0 register 2 | RSCAN1RMDF02 | <RSCAN1_base> + 0628$_H$ | RSCAN1 Global |
| | Receive buffer data field 1 register 2 | RSCAN1RMDF12 | <RSCAN1_base> + 062C$_H$ | RSCAN1 Global |
| | Receive buffer ID register 3 | RSCAN1RMID3 | <RSCAN1_base> + 0630$_H$ | RSCAN1 Global |
| | Receive buffer pointer register 3 | RSCAN1RMPTR3 | <RSCAN1_base> + 0634$_H$ | RSCAN1 Global |
| | Receive buffer data field 0 register 3 | RSCAN1RMDF03 | <RSCAN1_base> + 0638$_H$ | RSCAN1 Global |
| | Receive buffer data field 1 register 3 | RSCAN1RMDF13 | <RSCAN1_base> + 063C$_H$ | RSCAN1 Global |
| | Receive buffer ID register 4 | RSCAN1RMID4 | <RSCAN1_base> + 0640$_H$ | RSCAN1 Global |
| | Receive buffer pointer register 4 | RSCAN1RMPTR4 | <RSCAN1_base> + 0644$_H$ | RSCAN1 Global |

Table 21.213 Registers (5/9)

| Module | Register | Symbol | Address | Guard Group |
|---|---|---|---|---|
| RSCAN1 | Receive buffer data field 0 register 4 | RSCAN1RMDF04 | <RSCAN1_base> + $0648_H$ | RSCAN1 Global |
| | Receive buffer data field 1 register 4 | RSCAN1RMDF14 | <RSCAN1_base> + $064C_H$ | RSCAN1 Global |
| | Receive buffer ID register 5 | RSCAN1RMID5 | <RSCAN1_base> + $0650_H$ | RSCAN1 Global |
| | Receive buffer pointer register 5 | RSCAN1RMPTR5 | <RSCAN1_base> + $0654_H$ | RSCAN1 Global |
| | Receive buffer data field 0 register 5 | RSCAN1RMDF05 | <RSCAN1_base> + $0658_H$ | RSCAN1 Global |
| | Receive buffer data field 1 register 5 | RSCAN1RMDF15 | <RSCAN1_base> + $065C_H$ | RSCAN1 Global |
| | Receive buffer ID register 6 | RSCAN1RMID6 | <RSCAN1_base> + $0660_H$ | RSCAN1 Global |
| | Receive buffer pointer register 6 | RSCAN1RMPTR6 | <RSCAN1_base> + $0664_H$ | RSCAN1 Global |
| | Receive buffer data field 0 register 6 | RSCAN1RMDF06 | <RSCAN1_base> + $0668_H$ | RSCAN1 Global |
| | Receive buffer data field 1 register 6 | RSCAN1RMDF16 | <RSCAN1_base> + $066C_H$ | RSCAN1 Global |
| | Receive buffer ID register 7 | RSCAN1RMID7 | <RSCAN1_base> + $0670_H$ | RSCAN1 Global |
| | Receive buffer pointer register 7 | RSCAN1RMPTR7 | <RSCAN1_base> + $0674_H$ | RSCAN1 Global |
| | Receive buffer data field 0 register 7 | RSCAN1RMDF07 | <RSCAN1_base> + $0678_H$ | RSCAN1 Global |
| | Receive buffer data field 1 register 7 | RSCAN1RMDF17 | <RSCAN1_base> + $067C_H$ | RSCAN1 Global |
| | Receive buffer ID register 8 | RSCAN1RMID8 | <RSCAN1_base> + $0680_H$ | RSCAN1 Global |
| | Receive buffer pointer register 8 | RSCAN1RMPTR8 | <RSCAN1_base> + $0684_H$ | RSCAN1 Global |
| | Receive buffer data field 0 register 8 | RSCAN1RMDF08 | <RSCAN1_base> + $0688_H$ | RSCAN1 Global |
| | Receive buffer data field 1 register 8 | RSCAN1RMDF18 | <RSCAN1_base> + $068C_H$ | RSCAN1 Global |
| | Receive buffer ID register 9 | RSCAN1RMID9 | <RSCAN1_base> + $0690_H$ | RSCAN1 Global |
| | Receive buffer pointer register 9 | RSCAN1RMPTR9 | <RSCAN1_base> + $0694_H$ | RSCAN1 Global |
| | Receive buffer data field 0 register 9 | RSCAN1RMDF09 | <RSCAN1_base> + $0698_H$ | RSCAN1 Global |
| | Receive buffer data field 1 register 9 | RSCAN1RMDF19 | <RSCAN1_base> + $069C_H$ | RSCAN1 Global |
| | Receive buffer ID register 10 | RSCAN1RMID10 | <RSCAN1_base> + $06A0_H$ | RSCAN1 Global |
| | Receive buffer pointer register 10 | RSCAN1RMPTR10 | <RSCAN1_base> + $06A4_H$ | RSCAN1 Global |
| | Receive buffer data field 0 register 10 | RSCAN1RMDF010 | <RSCAN1_base> + $06A8_H$ | RSCAN1 Global |
| | Receive buffer data field 1 register 10 | RSCAN1RMDF110 | <RSCAN1_base> + $06AC_H$ | RSCAN1 Global |
| | Receive buffer ID register 11 | RSCAN1RMID11 | <RSCAN1_base> + $06B0_H$ | RSCAN1 Global |
| | Receive buffer pointer register 11 | RSCAN1RMPTR11 | <RSCAN1_base> + $06B4_H$ | RSCAN1 Global |
| | Receive buffer data field 0 register 11 | RSCAN1RMDF011 | <RSCAN1_base> + $06B8_H$ | RSCAN1 Global |
| | Receive buffer data field 1 register 11 | RSCAN1RMDF111 | <RSCAN1_base> + $06BC_H$ | RSCAN1 Global |
| | Receive buffer ID register 12 | RSCAN1RMID12 | <RSCAN1_base> + $06C0_H$ | RSCAN1 Global |
| | Receive buffer pointer register 12 | RSCAN1RMPTR12 | <RSCAN1_base> + $06C4_H$ | RSCAN1 Global |
| | Receive buffer data field 0 register 12 | RSCAN1RMDF012 | <RSCAN1_base> + $06C8_H$ | RSCAN1 Global |
| | Receive buffer data field 1 register 12 | RSCAN1RMDF112 | <RSCAN1_base> + $06CC_H$ | RSCAN1 Global |
| | Receive buffer ID register 13 | RSCAN1RMID13 | <RSCAN1_base> + $06D0_H$ | RSCAN1 Global |
| | Receive buffer pointer register 13 | RSCAN1RMPTR13 | <RSCAN1_base> + $06D4_H$ | RSCAN1 Global |
| | Receive buffer data field 0 register 13 | RSCAN1RMDF013 | <RSCAN1_base> + $06D8_H$ | RSCAN1 Global |
| | Receive buffer data field 1 register 13 | RSCAN1RMDF113 | <RSCAN1_base> + $06DC_H$ | RSCAN1 Global |
| | Receive buffer ID register 14 | RSCAN1RMID14 | <RSCAN1_base> + $06E0_H$ | RSCAN1 Global |
| | Receive buffer pointer register 14 | RSCAN1RMPTR14 | <RSCAN1_base> + $06E4_H$ | RSCAN1 Global |
| | Receive buffer data field 0 register 14 | RSCAN1RMDF014 | <RSCAN1_base> + $06E8_H$ | RSCAN1 Global |
| | Receive buffer data field 1 register 14 | RSCAN1RMDF114 | <RSCAN1_base> + $06EC_H$ | RSCAN1 Global |
| | Receive buffer ID register 15 | RSCAN1RMID15 | <RSCAN1_base> + $06F0_H$ | RSCAN1 Global |
| | Receive buffer pointer register 15 | RSCAN1RMPTR15 | <RSCAN1_base> + $06F4_H$ | RSCAN1 Global |
| | Receive buffer data field 0 register 15 | RSCAN1RMDF015 | <RSCAN1_base> + $06F8_H$ | RSCAN1 Global |
| | Receive buffer data field 1 register 15 | RSCAN1RMDF115 | <RSCAN1_base> + $06FC_H$ | RSCAN1 Global |
| | Receive FIFO buffer access ID register 0 | RSCAN1RFID0 | <RSCAN1_base> + $0E00_H$ | RSCAN1 Global |

Table 21.213 Registers (6/9)

| Module | Register | Symbol | Address | Guard Group |
|---|---|---|---|---|
| RSCAN1 | Receive FIFO buffer access pointer register 0 | RSCAN1RFPTR0 | <RSCAN1_base> + $0E04_H$ | RSCAN1 Global |
| | Receive FIFO buffer access data field 0 register 0 | RSCAN1RFDF00 | <RSCAN1_base> + $0E08_H$ | RSCAN1 Global |
| | Receive FIFO buffer access data field 1 register 0 | RSCAN1RFDF10 | <RSCAN1_base> + $0E0C_H$ | RSCAN1 Global |
| | Receive FIFO buffer access ID register 1 | RSCAN1RFID1 | <RSCAN1_base> + $0E10_H$ | RSCAN1 Global |
| | Receive FIFO buffer access pointer register 1 | RSCAN1RFPTR1 | <RSCAN1_base> + $0E14_H$ | RSCAN1 Global |
| | Receive FIFO buffer access data field 0 register 1 | RSCAN1RFDF01 | <RSCAN1_base> + $0E18_H$ | RSCAN1 Global |
| | Receive FIFO buffer access data field 1 register 1 | RSCAN1RFDF11 | <RSCAN1_base> + $0E1C_H$ | RSCAN1 Global |
| | Receive FIFO buffer access ID register 2 | RSCAN1RFID2 | <RSCAN1_base> + $0E20_H$ | RSCAN1 Global |
| | Receive FIFO buffer access pointer register 2 | RSCAN1RFPTR2 | <RSCAN1_base> + $0E24_H$ | RSCAN1 Global |
| | Receive FIFO buffer access data field 0 register 2 | RSCAN1RFDF02 | <RSCAN1_base> + $0E28_H$ | RSCAN1 Global |
| | Receive FIFO buffer access data field 1 register 2 | RSCAN1RFDF12 | <RSCAN1_base> + $0E2C_H$ | RSCAN1 Global |
| | Receive FIFO buffer access ID register 3 | RSCAN1RFID3 | <RSCAN1_base> + $0E30_H$ | RSCAN1 Global |
| | Receive FIFO buffer access pointer register 3 | RSCAN1RFPTR3 | <RSCAN1_base> + $0E34_H$ | RSCAN1 Global |
| | Receive FIFO buffer access data field 0 register 3 | RSCAN1RFDF03 | <RSCAN1_base> + $0E38_H$ | RSCAN1 Global |
| | Receive FIFO buffer access data field 1 register 3 | RSCAN1RFDF13 | <RSCAN1_base> + $0E3C_H$ | RSCAN1 Global |
| | Receive FIFO buffer access ID register 4 | RSCAN1RFID4 | <RSCAN1_base> + $0E40_H$ | RSCAN1 Global |
| | Receive FIFO buffer access pointer register 4 | RSCAN1RFPTR4 | <RSCAN1_base> + $0E44_H$ | RSCAN1 Global |
| | Receive FIFO buffer access data field 0 register 4 | RSCAN1RFDF04 | <RSCAN1_base> + $0E48_H$ | RSCAN1 Global |
| | Receive FIFO buffer access data field 1 register 4 | RSCAN1RFDF14 | <RSCAN1_base> + $0E4C_H$ | RSCAN1 Global |
| | Receive FIFO buffer access ID register 5 | RSCAN1RFID5 | <RSCAN1_base> + $0E50_H$ | RSCAN1 Global |
| | Receive FIFO buffer access pointer register 5 | RSCAN1RFPTR5 | <RSCAN1_base> + $0E54_H$ | RSCAN1 Global |
| | Receive FIFO buffer access data field 0 register 5 | RSCAN1RFDF05 | <RSCAN1_base> + $0E58_H$ | RSCAN1 Global |
| | Receive FIFO buffer access data field 1 register 5 | RSCAN1RFDF15 | <RSCAN1_base> + $0E5C_H$ | RSCAN1 Global |
| | Receive FIFO buffer access ID register 6 | RSCAN1RFID6 | <RSCAN1_base> + $0E60_H$ | RSCAN1 Global |
| | Receive FIFO buffer access pointer register 6 | RSCAN1RFPTR6 | <RSCAN1_base> + $0E64_H$ | RSCAN1 Global |
| | Receive FIFO buffer access data field 0 register 6 | RSCAN1RFDF06 | <RSCAN1_base> + $0E68_H$ | RSCAN1 Global |
| | Receive FIFO buffer access data field 1 register 6 | RSCAN1RFDF16 | <RSCAN1_base> + $0E6C_H$ | RSCAN1 Global |
| | Receive FIFO buffer access ID register 7 | RSCAN1RFID7 | <RSCAN1_base> + $0E70_H$ | RSCAN1 Global |
| | Receive FIFO buffer access pointer register 7 | RSCAN1RFPTR7 | <RSCAN1_base> + $0E74_H$ | RSCAN1 Global |
| | Receive FIFO buffer access data field 0 register 7 | RSCAN1RFDF07 | <RSCAN1_base> + $0E78_H$ | RSCAN1 Global |
| | Receive FIFO buffer access data field 1 register 7 | RSCAN1RFDF17 | <RSCAN1_base> + $0E7C_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer access ID register 0 | RSCAN1CFID0 | <RSCAN1_base> + $0E80_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer access pointer register 0 | RSCAN1CFPTR0 | <RSCAN1_base> + $0E84_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer access data field 0 register 0 | RSCAN1CFDF00 | <RSCAN1_base> + $0E88_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer access data field 1 register 0 | RSCAN1CFDF10 | <RSCAN1_base> + $0E8C_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer access ID register 1 | RSCAN1CFID1 | <RSCAN1_base> + $0E90_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer access pointer register 1 | RSCAN1CFPTR1 | <RSCAN1_base> + $0E94_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer access data field 0 register 1 | RSCAN1CFDF01 | <RSCAN1_base> + $0E98_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer access data field 1 register 1 | RSCAN1CFDF11 | <RSCAN1_base> + $0E9C_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer access ID register 2 | RSCAN1CFID2 | <RSCAN1_base> + $0EA0_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer access pointer register 2 | RSCAN1CFPTR2 | <RSCAN1_base> + $0EA4_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer access data field 0 register 2 | RSCAN1CFDF02 | <RSCAN1_base> + $0EA8_H$ | RSCAN1 Global |
| | Transmit/receive FIFO buffer access data field 1 register 2 | RSCAN1CFDF12 | <RSCAN1_base> + $0EAC_H$ | RSCAN1 Global |
| | Transmit buffer ID register 0 | RSCAN1TMID0 | <RSCAN1_base> + $1000_H$ | RSCAN1 Global |
| | Transmit buffer pointer register 0 | RSCAN1TMPTR0 | <RSCAN1_base> + $1004_H$ | RSCAN1 Global |
| | Transmit buffer data field 0 register 0 | RSCAN1TMDF00 | <RSCAN1_base> + $1008_H$ | RSCAN1 Global |
| | Transmit buffer data field 1 register 0 | RSCAN1TMDF10 | <RSCAN1_base> + $100C_H$ | RSCAN1 Global |

Table 21.213 Registers (7/9)

| Module | Register | Symbol | Address | Guard Group |
|---|---|---|---|---|
| RSCAN1 | Transmit buffer ID register 1 | RSCAN1TMID1 | <RSCAN1_base> + $1010_H$ | RSCAN1 Global |
| | Transmit buffer pointer register 1 | RSCAN1TMPTR1 | <RSCAN1_base> + $1014_H$ | RSCAN1 Global |
| | Transmit buffer data field 0 register 1 | RSCAN1TMDF01 | <RSCAN1_base> + $1018_H$ | RSCAN1 Global |
| | Transmit buffer data field 1 register 1 | RSCAN1TMDF11 | <RSCAN1_base> + $101C_H$ | RSCAN1 Global |
| | Transmit buffer ID register 2 | RSCAN1TMID2 | <RSCAN1_base> + $1020_H$ | RSCAN1 Global |
| | Transmit buffer pointer register 2 | RSCAN1TMPTR2 | <RSCAN1_base> + $1024_H$ | RSCAN1 Global |
| | Transmit buffer data field 0 register 2 | RSCAN1TMDF02 | <RSCAN1_base> + $1028_H$ | RSCAN1 Global |
| | Transmit buffer data field 1 register 2 | RSCAN1TMDF12 | <RSCAN1_base> + $102C_H$ | RSCAN1 Global |
| | Transmit buffer ID register 3 | RSCAN1TMID3 | <RSCAN1_base> + $1030_H$ | RSCAN1 Global |
| | Transmit buffer pointer register 3 | RSCAN1TMPTR3 | <RSCAN1_base> + $1034_H$ | RSCAN1 Global |
| | Transmit buffer data field 0 register 3 | RSCAN1TMDF03 | <RSCAN1_base> + $1038_H$ | RSCAN1 Global |
| | Transmit buffer data field 1 register 3 | RSCAN1TMDF13 | <RSCAN1_base> + $103C_H$ | RSCAN1 Global |
| | Transmit buffer ID register 4 | RSCAN1TMID4 | <RSCAN1_base> + $1040_H$ | RSCAN1 Global |
| | Transmit buffer pointer register 4 | RSCAN1TMPTR4 | <RSCAN1_base> + $1044_H$ | RSCAN1 Global |
| | Transmit buffer data field 0 register 4 | RSCAN1TMDF04 | <RSCAN1_base> + $1048_H$ | RSCAN1 Global |
| | Transmit buffer data field 1 register 4 | RSCAN1TMDF14 | <RSCAN1_base> + $104C_H$ | RSCAN1 Global |
| | Transmit buffer ID register 5 | RSCAN1TMID5 | <RSCAN1_base> + $1050_H$ | RSCAN1 Global |
| | Transmit buffer pointer register 5 | RSCAN1TMPTR5 | <RSCAN1_base> + $1054_H$ | RSCAN1 Global |
| | Transmit buffer data field 0 register 5 | RSCAN1TMDF05 | <RSCAN1_base> + $1058_H$ | RSCAN1 Global |
| | Transmit buffer data field 1 register 5 | RSCAN1TMDF15 | <RSCAN1_base> + $105C_H$ | RSCAN1 Global |
| | Transmit buffer ID register 6 | RSCAN1TMID6 | <RSCAN1_base> + $1060_H$ | RSCAN1 Global |
| | Transmit buffer pointer register 6 | RSCAN1TMPTR6 | <RSCAN1_base> + $1064_H$ | RSCAN1 Global |
| | Transmit buffer data field 0 register 6 | RSCAN1TMDF06 | <RSCAN1_base> + $1068_H$ | RSCAN1 Global |
| | Transmit buffer data field 1 register 6 | RSCAN1TMDF16 | <RSCAN1_base> + $106C_H$ | RSCAN1 Global |
| | Transmit buffer ID register 7 | RSCAN1TMID7 | <RSCAN1_base> + $1070_H$ | RSCAN1 Global |
| | Transmit buffer pointer register 7 | RSCAN1TMPTR7 | <RSCAN1_base> + $1074_H$ | RSCAN1 Global |
| | Transmit buffer data field 0 register 7 | RSCAN1TMDF07 | <RSCAN1_base> + $1078_H$ | RSCAN1 Global |
| | Transmit buffer data field 1 register 7 | RSCAN1TMDF17 | <RSCAN1_base> + $107C_H$ | RSCAN1 Global |
| | Transmit buffer ID register 8 | RSCAN1TMID8 | <RSCAN1_base> + $1080_H$ | RSCAN1 Global |
| | Transmit buffer pointer register 8 | RSCAN1TMPTR8 | <RSCAN1_base> + $1084_H$ | RSCAN1 Global |
| | Transmit buffer data field 0 register 8 | RSCAN1TMDF08 | <RSCAN1_base> + $1088_H$ | RSCAN1 Global |
| | Transmit buffer data field 1 register 8 | RSCAN1TMDF18 | <RSCAN1_base> + $108C_H$ | RSCAN1 Global |
| | Transmit buffer ID register 9 | RSCAN1TMID9 | <RSCAN1_base> + $1090_H$ | RSCAN1 Global |
| | Transmit buffer pointer register 9 | RSCAN1TMPTR9 | <RSCAN1_base> + $1094_H$ | RSCAN1 Global |
| | Transmit buffer data field 0 register 9 | RSCAN1TMDF09 | <RSCAN1_base> + $1098_H$ | RSCAN1 Global |
| | Transmit buffer data field 1 register 9 | RSCAN1TMDF19 | <RSCAN1_base> + $109C_H$ | RSCAN1 Global |
| | Transmit buffer ID register 10 | RSCAN1TMID10 | <RSCAN1_base> + $10A0_H$ | RSCAN1 Global |
| | Transmit buffer pointer register 10 | RSCAN1TMPTR10 | <RSCAN1_base> + $10A4_H$ | RSCAN1 Global |
| | Transmit buffer data field 0 register 10 | RSCAN1TMDF010 | <RSCAN1_base> + $10A8_H$ | RSCAN1 Global |
| | Transmit buffer data field 1 register 10 | RSCAN1TMDF110 | <RSCAN1_base> + $10AC_H$ | RSCAN1 Global |
| | Transmit buffer ID register 11 | RSCAN1TMID11 | <RSCAN1_base> + $10B0_H$ | RSCAN1 Global |
| | Transmit buffer pointer register 11 | RSCAN1TMPTR11 | <RSCAN1_base> + $10B4_H$ | RSCAN1 Global |
| | Transmit buffer data field 0 register 11 | RSCAN1TMDF011 | <RSCAN1_base> + $10B8_H$ | RSCAN1 Global |
| | Transmit buffer data field 1 register 11 | RSCAN1TMDF111 | <RSCAN1_base> + $10BC_H$ | RSCAN1 Global |
| | Transmit buffer ID register 12 | RSCAN1TMID12 | <RSCAN1_base> + $10C0_H$ | RSCAN1 Global |
| | Transmit buffer pointer register 12 | RSCAN1TMPTR12 | <RSCAN1_base> + $10C4_H$ | RSCAN1 Global |
| | Transmit buffer data field 0 register 12 | RSCAN1TMDF012 | <RSCAN1_base> + $10C8_H$ | RSCAN1 Global |

Table 21.213 Registers (8/9)

| Module | Register | Symbol | Address | Guard Group |
|---|---|---|---|---|
| RSCAN1 | Transmit buffer data field 1 register 12 | RSCAN1TMDF112 | <RSCAN1_base> + $10CC_H$ | RSCAN1 Global |
| | Transmit buffer ID register 13 | RSCAN1TMID13 | <RSCAN1_base> + $10D0_H$ | RSCAN1 Global |
| | Transmit buffer pointer register 13 | RSCAN1TMPTR13 | <RSCAN1_base> + $10D4_H$ | RSCAN1 Global |
| | Transmit buffer data field 0 register 13 | RSCAN1TMDF013 | <RSCAN1_base> + $10D8_H$ | RSCAN1 Global |
| | Transmit buffer data field 1 register 13 | RSCAN1TMDF113 | <RSCAN1_base> + $10DC_H$ | RSCAN1 Global |
| | Transmit buffer ID register 14 | RSCAN1TMID14 | <RSCAN1_base> + $10E0_H$ | RSCAN1 Global |
| | Transmit buffer pointer register 14 | RSCAN1TMPTR14 | <RSCAN1_base> + $10E4_H$ | RSCAN1 Global |
| | Transmit buffer data field 0 register 14 | RSCAN1TMDF014 | <RSCAN1_base> + $10E8_H$ | RSCAN1 Global |
| | Transmit buffer data field 1 register 14 | RSCAN1TMDF114 | <RSCAN1_base> + $10EC_H$ | RSCAN1 Global |
| | Transmit buffer ID register 15 | RSCAN1TMID15 | <RSCAN1_base> + $10F0_H$ | RSCAN1 Global |
| | Transmit buffer pointer register 15 | RSCAN1TMPTR15 | <RSCAN1_base> + $10F4_H$ | RSCAN1 Global |
| | Transmit buffer data field 0 register 15 | RSCAN1TMDF015 | <RSCAN1_base> + $10F8_H$ | RSCAN1 Global |
| | Transmit buffer data field 1 register 15 | RSCAN1TMDF115 | <RSCAN1_base> + $10FC_H$ | RSCAN1 Global |
| | Transmit history access register m | RSCAN1THLACCm | <RSCAN1_base> + $1800_H$ | RSCAN1 Ch0 |
| | RAM test page access register 0 | RSCAN1RPGACC0 | <RSCAN1_base> + $1900_H$ | RSCAN1 Global |
| | RAM test page access register 1 | RSCAN1RPGACC1 | <RSCAN1_base> + $1904_H$ | RSCAN1 Global |
| | RAM test page access register 2 | RSCAN1RPGACC2 | <RSCAN1_base> + $1908_H$ | RSCAN1 Global |
| | RAM test page access register 3 | RSCAN1RPGACC3 | <RSCAN1_base> + $190C_H$ | RSCAN1 Global |
| | RAM test page access register 4 | RSCAN1RPGACC4 | <RSCAN1_base> + $1910_H$ | RSCAN1 Global |
| | RAM test page access register 5 | RSCAN1RPGACC5 | <RSCAN1_base> + $1914_H$ | RSCAN1 Global |
| | RAM test page access register 6 | RSCAN1RPGACC6 | <RSCAN1_base> + $1918_H$ | RSCAN1 Global |
| | RAM test page access register 7 | RSCAN1RPGACC7 | <RSCAN1_base> + $191C_H$ | RSCAN1 Global |
| | RAM test page access register 8 | RSCAN1RPGACC8 | <RSCAN1_base> + $1920_H$ | RSCAN1 Global |
| | RAM test page access register 9 | RSCAN1RPGACC9 | <RSCAN1_base> + $1924_H$ | RSCAN1 Global |
| | RAM test page access register 10 | RSCAN1RPGACC10 | <RSCAN1_base> + $1928_H$ | RSCAN1 Global |
| | RAM test page access register 11 | RSCAN1RPGACC11 | <RSCAN1_base> + $192C_H$ | RSCAN1 Global |
| | RAM test page access register 12 | RSCAN1RPGACC12 | <RSCAN1_base> + $1930_H$ | RSCAN1 Global |
| | RAM test page access register 13 | RSCAN1RPGACC13 | <RSCAN1_base> + $1934_H$ | RSCAN1 Global |
| | RAM test page access register 14 | RSCAN1RPGACC14 | <RSCAN1_base> + $1938_H$ | RSCAN1 Global |
| | RAM test page access register 15 | RSCAN1RPGACC15 | <RSCAN1_base> + $193C_H$ | RSCAN1 Global |
| | RAM test page access register 16 | RSCAN1RPGACC16 | <RSCAN1_base> + $1940_H$ | RSCAN1 Global |
| | RAM test page access register 17 | RSCAN1RPGACC17 | <RSCAN1_base> + $1944_H$ | RSCAN1 Global |
| | RAM test page access register 18 | RSCAN1RPGACC18 | <RSCAN1_base> + $1948_H$ | RSCAN1 Global |
| | RAM test page access register 19 | RSCAN1RPGACC19 | <RSCAN1_base> + $194C_H$ | RSCAN1 Global |
| | RAM test page access register 20 | RSCAN1RPGACC20 | <RSCAN1_base> + $1950_H$ | RSCAN1 Global |
| | RAM test page access register 21 | RSCAN1RPGACC21 | <RSCAN1_base> + $1954_H$ | RSCAN1 Global |
| | RAM test page access register 22 | RSCAN1RPGACC22 | <RSCAN1_base> + $1958_H$ | RSCAN1 Global |
| | RAM test page access register 23 | RSCAN1RPGACC23 | <RSCAN1_base> + $195C_H$ | RSCAN1 Global |
| | RAM test page access register 24 | RSCAN1RPGACC24 | <RSCAN1_base> + $1960_H$ | RSCAN1 Global |
| | RAM test page access register 25 | RSCAN1RPGACC25 | <RSCAN1_base> + $1964_H$ | RSCAN1 Global |
| | RAM test page access register 26 | RSCAN1RPGACC26 | <RSCAN1_base> + $1968_H$ | RSCAN1 Global |
| | RAM test page access register 27 | RSCAN1RPGACC27 | <RSCAN1_base> + $196C_H$ | RSCAN1 Global |
| | RAM test page access register 28 | RSCAN1RPGACC28 | <RSCAN1_base> + $1970_H$ | RSCAN1 Global |
| | RAM test page access register 29 | RSCAN1RPGACC29 | <RSCAN1_base> + $1974_H$ | RSCAN1 Global |
| | RAM test page access register 30 | RSCAN1RPGACC30 | <RSCAN1_base> + $1978_H$ | RSCAN1 Global |
| | RAM test page access register 31 | RSCAN1RPGACC31 | <RSCAN1_base> + $197C_H$ | RSCAN1 Global |
| | RAM test page access register 32 | RSCAN1RPGACC32 | <RSCAN1_base> + $1980_H$ | RSCAN1 Global |

Table 21.213 Registers (9/9)

| Module | Register | Symbol | Address | Guard Group |
|---|---|---|---|---|
| RSCAN1 | RAM test page access register 33 | RSCAN1RPGACC33 | <RSCAN1_base> + $1984_H$ | RSCAN1 Global |
| | RAM test page access register 34 | RSCAN1RPGACC34 | <RSCAN1_base> + $1988_H$ | RSCAN1 Global |
| | RAM test page access register 35 | RSCAN1RPGACC35 | <RSCAN1_base> + $198C_H$ | RSCAN1 Global |
| | RAM test page access register 36 | RSCAN1RPGACC36 | <RSCAN1_base> + $1990_H$ | RSCAN1 Global |
| | RAM test page access register 37 | RSCAN1RPGACC37 | <RSCAN1_base> + $1994_H$ | RSCAN1 Global |
| | RAM test page access register 38 | RSCAN1RPGACC38 | <RSCAN1_base> + $1998_H$ | RSCAN1 Global |
| | RAM test page access register 39 | RSCAN1RPGACC39 | <RSCAN1_base> + $199C_H$ | RSCAN1 Global |
| | RAM test page access register 40 | RSCAN1RPGACC40 | <RSCAN1_base> + $19A0_H$ | RSCAN1 Global |
| | RAM test page access register 41 | RSCAN1RPGACC41 | <RSCAN1_base> + $19A4_H$ | RSCAN1 Global |
| | RAM test page access register 42 | RSCAN1RPGACC42 | <RSCAN1_base> + $19A8_H$ | RSCAN1 Global |
| | RAM test page access register 43 | RSCAN1RPGACC43 | <RSCAN1_base> + $19AC_H$ | RSCAN1 Global |
| | RAM test page access register 44 | RSCAN1RPGACC44 | <RSCAN1_base> + $19B0_H$ | RSCAN1 Global |
| | RAM test page access register 45 | RSCAN1RPGACC45 | <RSCAN1_base> + $19B4_H$ | RSCAN1 Global |
| | RAM test page access register 46 | RSCAN1RPGACC46 | <RSCAN1_base> + $19B8_H$ | RSCAN1 Global |
| | RAM test page access register 47 | RSCAN1RPGACC47 | <RSCAN1_base> + $19BC_H$ | RSCAN1 Global |
| | RAM test page access register 48 | RSCAN1RPGACC48 | <RSCAN1_base> + $19C0_H$ | RSCAN1 Global |
| | RAM test page access register 49 | RSCAN1RPGACC49 | <RSCAN1_base> + $19C4_H$ | RSCAN1 Global |
| | RAM test page access register 50 | RSCAN1RPGACC50 | <RSCAN1_base> + $19C8_H$ | RSCAN1 Global |
| | RAM test page access register 51 | RSCAN1RPGACC51 | <RSCAN1_base> + $19CC_H$ | RSCAN1 Global |
| | RAM test page access register 52 | RSCAN1RPGACC52 | <RSCAN1_base> + $19D0_H$ | RSCAN1 Global |
| | RAM test page access register 53 | RSCAN1RPGACC53 | <RSCAN1_base> + $19D4_H$ | RSCAN1 Global |
| | RAM test page access register 54 | RSCAN1RPGACC54 | <RSCAN1_base> + $19D8_H$ | RSCAN1 Global |
| | RAM test page access register 55 | RSCAN1RPGACC55 | <RSCAN1_base> + $19DC_H$ | RSCAN1 Global |
| | RAM test page access register 56 | RSCAN1RPGACC56 | <RSCAN1_base> + $19E0_H$ | RSCAN1 Global |
| | RAM test page access register 57 | RSCAN1RPGACC57 | <RSCAN1_base> + $19E4_H$ | RSCAN1 Global |
| | RAM test page access register 58 | RSCAN1RPGACC58 | <RSCAN1_base> + $19E8_H$ | RSCAN1 Global |
| | RAM test page access register 59 | RSCAN1RPGACC59 | <RSCAN1_base> + $19EC_H$ | RSCAN1 Global |
| | RAM test page access register 60 | RSCAN1RPGACC60 | <RSCAN1_base> + $19F0_H$ | RSCAN1 Global |
| | RAM test page access register 61 | RSCAN1RPGACC61 | <RSCAN1_base> + $19F4_H$ | RSCAN1 Global |
| | RAM test page access register 62 | RSCAN1RPGACC62 | <RSCAN1_base> + $19F8_H$ | RSCAN1 Global |
| | RAM test page access register 63 | RSCAN1RPGACC63 | <RSCAN1_base> + $19FC_H$ | RSCAN1 Global |

**Note:** For details on the guard group, see **Section 33, Functional Safety**.

**Table 21.214 Transmit Buffer p Allocated to Each Channel**

| | CANm |
|---|---|
| Transmit buffer p | Transmit buffer 16 × i + 0 |
| | Transmit buffer 16 × i + 1 |
| | Transmit buffer 16 × i + 2 |
| | Transmit buffer 16 × i + 3 |
| | Transmit buffer 16 × i + 4 |
| | Transmit buffer 16 × i + 5 |
| | Transmit buffer 16 × i + 6 |
| | Transmit buffer 16 × i + 7 |
| | Transmit buffer 16 × i + 8 |
| | Transmit buffer 16 × i + 9 |
| | Transmit buffer 16 × i + 10 |
| | Transmit buffer 16 × i + 11 |
| | Transmit buffer 16 × i + 12 |
| | Transmit buffer 16 × i + 13 |
| | Transmit buffer 16 × i + 14 |
| | Transmit buffer 16 × i + 15 |

**Table 21.215 Transmit/Receive FIFO Buffer k Allocated to Each Channel**

| | CANm |
|---|---|
| Transmit/receive FIFO buffer k | Transmit/receive FIFO buffer 3 × i + 0 |
| | Transmit/receive FIFO buffer 3 × i + 1 |
| | Transmit/receive FIFO buffer 3 × i + 2 |

**Table 21.216 Transmit Buffer p Linked to the Transmit/Receive FIFO Buffer by the Setting of Bits CFTML[3:0]**

| Setting of Bits CFTML[3:0] | Transmit Buffer p Linked to the Transmit/Receive FIFO Buffer |
|---|---|
| $0000_B$ | Transmit buffer 16 × i + 0 |
| $0001_B$ | Transmit buffer 16 × i + 1 |
| $0010_B$ | Transmit buffer 16 × i + 2 |
| $0011_B$ | Transmit buffer 16 × i + 3 |
| $0100_B$ | Transmit buffer 16 × i + 4 |
| $0101_B$ | Transmit buffer 16 × i + 5 |
| $0110_B$ | Transmit buffer 16 × i + 6 |
| $0111_B$ | Transmit buffer 16 × i + 7 |
| $1000_B$ | Transmit buffer 16 × i + 8 |
| $1001_B$ | Transmit buffer 16 × i + 9 |
| $1010_B$ | Transmit buffer 16 × i + 10 |
| $1011_B$ | Transmit buffer 16 × i + 11 |
| $1100_B$ | Transmit buffer 16 × i + 12 |
| $1101_B$ | Transmit buffer 16 × i + 13 |
| $1110_B$ | Transmit buffer 16 × i + 14 |
| $1111_B$ | Transmit buffer 16 × i + 15 |

**Table 21.217 Transmit Buffer p Allocated to the Transmit Queue of Each Channel**

| Setting of Bits TXQDC[3:0] | Transmit Buffer p Allocated to the Transmit Queue |
|---|---|
| $0000_B$ | Setting prohibited |
| $0001_B$ | Setting prohibited |
| $0010_B$ | Transmit buffer 16 × i + 15 to 16 × i + 13 |
| $0011_B$ | Transmit buffer 16 × i + 15 to 16 × i + 12 |
| $0100_B$ | Transmit buffer 16 × i + 15 to 16 × i + 11 |
| $0101_B$ | Transmit buffer 16 × i + 15 to 16 × i + 10 |
| $0110_B$ | Transmit buffer 16 × i + 15 to 16 × i + 9 |
| $0111_B$ | Transmit buffer 16 × i + 15 to 16 × i + 8 |
| $1000_B$ | Transmit buffer 16 × i + 15 to 16 × i + 7 |
| $1001_B$ | Transmit buffer 16 × i + 15 to 16 × i + 6 |
| $1010_B$ | Transmit buffer 16 × i + 15 to 16 × i + 5 |
| $1011_B$ | Transmit buffer 16 × i + 15 to 16 × i + 4 |
| $1100_B$ | Transmit buffer 16 × i + 15 to 16 × i + 3 |
| $1101_B$ | Transmit buffer 16 × i + 15 to 16 × i + 2 |
| $1110_B$ | Transmit buffer 16 × i + 15 to 16 × i + 1 |
| $1111_B$ | Transmit buffer 16 × i + 15 to 16 × i + 0 |

## 21.16.2 RSCAN1CmCFG — Channel Configuration Register (m = 6)

**Access:** RSCAN1CmCFG register can be read or written in 32-bit units.
RSCAN1CmCFGL, RSCAN1CmCFGH registers can be read or written in 16-bit units.
RSCAN1CmCFGLL, RSCAN1CmCFGLH, RSCAN1CmCFGHL, RSCAN1CmCFGHH registers can be read or written in 8-bit units.

**Address:** RSCAN1CmCFG: <RSCAN1_base> + $0000_H$ + ($10_H$ × i)

RSCAN1CmCFGL: <RSCAN1_base> + $0000_H$ + ($10_H$ × i),
RSCAN1CmCFGH: <RSCAN1_base> + $0002_H$ + ($10_H$ × i)

RSCAN1CmCFGLL: <RSCAN1_base> + $0000_H$ + ($10_H$ × i),
RSCAN1CmCFGLH: <RSCAN1_base> + $0001_H$ + ($10_H$ × i),
RSCAN1CmCFGHL: <RSCAN1_base> + $0002_H$ + ($10_H$ × i),
RSCAN1CmCFGHH: <RSCAN1_base> + $0003_H$ + ($10_H$ × i)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | SJW[1:0] | | — | TSEG2[2:0] | | | TSEG1[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | BRP[9:0] | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.218 RSCAN1CmCFG Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 25, 24 | SJW[1:0] | Resynchronization Jump Width Control<br>b25 b24<br>0 0: 1 Tq<br>0 1: 2 Tq<br>1 0: 3 Tq<br>1 1: 4 Tq |
| 23 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 22 to 20 | TSEG2[2:0] | Time Segment 2 Control<br>b22 b21 b20<br>0 0 0: Setting prohibited<br>0 0 1: 2 Tq<br>0 1 0: 3 Tq<br>0 1 1: 4 Tq<br>1 0 0: 5 Tq<br>1 0 1: 6 Tq<br>1 1 0: 7 Tq<br>1 1 1: 8 Tq |

**Table 21.218 RSCAN1CmCFG Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 19 to 16 | TSEG1[3:0] | Time Segment 1 Control<br>b19 b18 b17 b16<br>0 0 0 0: Setting prohibited<br>0 0 0 1: Setting prohibited<br>0 0 1 0: Setting prohibited<br>0 0 1 1: 4 Tq<br>0 1 0 0: 5 Tq<br>0 1 0 1: 6 Tq<br>0 1 1 0: 7 Tq<br>0 1 1 1: 8 Tq<br>1 0 0 0: 9 Tq<br>1 0 0 1: 10 Tq<br>1 0 1 0: 11 Tq<br>1 0 1 1: 12 Tq<br>1 1 0 0: 13 Tq<br>1 1 0 1: 14 Tq<br>1 1 1 0: 15 Tq<br>1 1 1 1: 16 Tq |
| 15 to 10 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 9 to 0 | BRP[9:0] | Prescaler Division Ratio Setting<br>When these bits are set to P (0 to 1023), the baud rate prescaler divides fCAN by P + 1. |

Modify the RSCAN1CmCFG register in channel reset mode or channel halt mode. Set this register in channel reset mode before transitioning to channel communication mode or channel halt mode. For a description of the bit timing parameters and settings, see **Section 21.22.1, Initial Settings**.

### SJW[1:0] Bits

These bits are used to specify a Tq value for the resynchronization jump width.Values from 1Tq to 4Tq can be specified.
Set a value less than or equal to the value of the TSEG2 bits.

### TSEG2[2:0] Bits

These bits are used to specify a Tq value for the length of phase segment 2 (PHASE_SEG2).
Values from 2Tq to 8Tq can be specified.
Set a value smaller than the value of the TSEG1 bits.

### TSEG1[3:0] Bits

These bits are used to specify a Tq value for the total length of the propagation segment (PROP_SEG) and phase segment 1 (PHASE_SEG1).
Values from 4Tq to 16Tq can be specified.

### BRP[9:0] Bits

The CANmTq clock (fCANTQm) is calculated by dividing the CAN clock (fCAN) by the baud rate prescaler, ((BRP[9:0]) + 1). One clock cycle of the CANmTq clock is 1 Time Quantum (Tq).

## 21.16.3 RSCAN1CmCTR — Channel Control Register (m = 6)

**Access:** RSCAN1CmCTR register can be read or written in 32-bit units.
RSCAN1CmCTRL, RSCAN1CmCTRH registers can be read or written in 16-bit units.
RSCAN1CmCTRLL, RSCAN1CmCTRLH, RSCAN1CmCTRHL, RSCAN1CmCTRHH registers can be read or written in 8-bit units.

**Address:** RSCAN1CmCTR: <RSCAN1_base> + $0004_H$ + ($10_H$ × i)

RSCAN1CmCTRL: <RSCAN1_base> + $0004_H$ + ($10_H$ × i),
RSCAN1CmCTRH: <RSCAN1_base> + $0006_H$ + ($10_H$ × i)

RSCAN1CmCTRLL: <RSCAN1_base> + $0004_H$ + ($10_H$ × i),
RSCAN1CmCTRLH: <RSCAN1_base> + $0005_H$ + ($10_H$ × i),
RSCAN1CmCTRHL: <RSCAN1_base> + $0006_H$ + ($10_H$ × i),
RSCAN1CmCTRHH: <RSCAN1_base> + $0007_H$ + ($10_H$ × i)

**Value after reset:** $0000\ 0005_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | CTMS[1:0] | | CTME | ERRD | BOM[1:0] | | — | — | — | — | TAIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ALIE | BLIE | OLIE | BORIE | BOEIE | EPIE | EWIE | BEIE | — | — | — | — | RTBO | CSLPR | CHMDC[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 21.219 RSCAN1CmCTR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 26, 25 | CTMS[1:0] | Communication Test Mode Select<br>b26 b25<br>0 0: Standard test mode<br>0 1: Listen-only mode<br>1 0: Self-test mode 0 (external loopback mode)<br>1 1: Self-test mode 1 (internal loopback mode) |
| 24 | CTME | Communication Test Mode Enable<br>0: Communication test mode is disabled.<br>1: Communication test mode is enabled. |
| 23 | ERRD | Error Display Mode Select<br>0: Error flags are displayed only for the first error information after bits 14 to 8 in RSCAN1CmERFL are all cleared.<br>1: Error flags for all error information are displayed. |
| 22, 21 | BOM[1:0] | Bus Off Recovery Mode Select<br>b22 b21<br>0 0: ISO11898-1 compliant<br>0 1: Entry to channel halt mode automatically at bus-off entry<br>1 0: Entry to channel halt mode automatically at bus-off end<br>1 1: Entry to channel halt mode (in bus-off state) by program request |
| 20 to 17 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 16 | TAIE | Transmit Abort Interrupt Enable<br>0: Transmit abort interrupt is disabled.<br>1: Transmit abort interrupt is enabled. |
| 15 | ALIE | Arbitration Lost Interrupt Enable<br>0: Arbitration lost interrupt is disabled.<br>1: Arbitration lost interrupt is enabled. |

Table 21.219 RSCAN1CmCTR Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 14 | BLIE | Bus Lock Interrupt Enable<br>0: Bus lock interrupt is disabled.<br>1: Bus lock interrupt is enabled. |
| 13 | OLIE | Overload Frame Transmit Interrupt Enable<br>0: Overload frame transmit interrupt is disabled.<br>1: Overload frame transmit interrupt is enabled. |
| 12 | BORIE | Bus Off Recovery Interrupt Enable<br>0: Bus off recovery interrupt is disabled.<br>1: Bus off recovery interrupt is enabled. |
| 11 | BOEIE | Bus Off Entry Interrupt Enable<br>0: Bus off entry interrupt is disabled.<br>1: Bus off entry interrupt is enabled. |
| 10 | EPIE | Error Passive Interrupt Enable<br>0: Error passive interrupt is disabled.<br>1: Error passive interrupt is enabled. |
| 9 | EWIE | Error Warning Interrupt Enable<br>0: Error warning interrupt is disabled.<br>1: Error warning interrupt is enabled. |
| 8 | BEIE | Bus Error Interrupt Enable<br>0: Bus error interrupt is disabled.<br>1: Bus error interrupt is enabled. |
| 7 to 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | RTBO | Forcible Return from Bus-off<br>When this bit is set to 1, forcible return from the bus off state is made. This bit is always read as 0. |
| 2 | CSLPR | Channel Stop Mode<br>0: Other than channel stop mode<br>1: Channel stop mode |
| 1, 0 | CHMDC[1:0] | Mode Select<br>b1 b0<br>0 0: Channel communication mode<br>0 1: Channel reset mode<br>1 0: Channel halt mode<br>1 1: Setting prohibited |

### CTMS[1:0] Bits

These bits are used to select a communication test mode. Modify these bits in channel halt mode only. These bits are set to 0 in channel reset mode.

### CTME Bit

Setting this bit to 1 enables communication test mode. Modify this bit in channel halt mode. This bit is set to 0 in channel reset mode.

### ERRD Bit

This bit is used to control the display mode of bits 14 to 8 in the RSCAN1CmERFL register.
When this bit is cleared to 0, only the flags of the first error event are set to 1. If two or more errors occur in the first error event, all the flags of the detected errors are set to 1.

When this bit is set to 1, all the flags of errors that have occurred are set to 1 regardless of the error occurrence order.

Modify this bit only in channel reset mode or channel halt mode.

#### BOM[1:0] Bits

These bits are used to select the bus off recovery mode of the RS-CAN module.

When the BOM[1:0] bits are set to $00_B$, return from the bus off state to the error active state is compliant with the CAN specifications. That is, the RS-CAN module reenters the CAN communication (error active state) after 11 consecutive recessive bits are detected 128 times. A bus off recovery interrupt request is generated at the time of return from the bus off state. Even if the CHMDC[1:0] bits are set to $10_B$ (channel halt mode) before recessive bits are detected 128 times, the RS-CAN module does not transition to channel halt mode until recessive bits are detected 128 times.

When the RS-CAN module reaches the bus off state when the BOM[1:0] bits are set to $01_B$, the CHMDC[1:0] bits in the RSCAN1CmCTR register (m = 6) are set to $10_B$ and the RS-CAN module transitions to channel halt mode. No bus off recovery interrupt request is generated at the time of return from the bus off state and the TEC[7:0] and REC[7:0] bits in the RSCAN1CmSTS register are cleared to $00_H$.

When the RS-CAN module reaches the bus off state when the BOM[1:0] bits are set to $10_B$, the CHMDC[1:0] bits are set to $10_B$ and the RS-CAN module transitions to channel halt mode after return from the bus off state (11 consecutive recessive bits are detected 128 times). A bus off recovery interrupt request is generated at the time of return from the bus off state and the TEC[7:0] and REC[7:0] bits are cleared to $00_H$.

When the BOM[1:0] bits are set to $11_B$ and the CHMDC[1:0] bits are set to $10_B$ while the RS-CAN module is in the bus off state, the RS-CAN module transitions to channel halt mode. No bus off recovery interrupt request is generated at the time of return from the bus off state and the TEC[7:0] and REC[7:0] bits are cleared to $00_H$. However, if 11 consecutive recessive bits are detected 128 times and the RS-CAN module has recovered to the error active state from the bus off state before the CHMDC[1:0] bits are set to $10_B$, a bus off recovery interrupt request is generated.

If a program writes to the CHMDC[1:0] bit at the same time as the RS-CAN module transition to channel halt mode (at bus off entry when the BOM[1:0] bits are $01_B$ or at bus off end when the BOM[1:0] bits are $10_B$), the program's writing takes precedence. Modify the BOM[1:0] bits only in channel reset mode.

#### TAIE Bit

When transmit abort of the transmit buffer is completed with the TAIE bit set to 1, an interrupt request is generated. Modify this bit only in channel reset mode.

#### ALIE Bit

When the ALF flag in the RSCAN1CmERFL register is set to 1 with the ALIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### BLIE Bit

When the BLF flag in the RSCAN1CmERFL register is set to 1 with the BLIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### OLIE Bit

When the OVLF flag in the RSCAN1CmERFL register is set to 1 with the OLIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### BORIE Bit

When the BORF flag in the RSCAN1CmERFL register is set to 1 with the BORIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### BOEIE Bit

When the BOEF flag in the RSCAN1CmERFL register is set to 1 with the BOEIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### EPIE Bit

When the EPF flag in the RSCAN1CmERFL register is set to 1 with the EPIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### EWIE Bit

When the EWF flag in the RSCAN1CmERFL register is set to 1 with the EWIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel reset mode.

#### BEIE Bit

When the BEF flag in the RSCAN1CmERFL register is set to 1 with the BEIE bit set to 1, an error interrupt request is generated. Modify this bit only in channel halt mode.

#### RTBO Bit

Setting this bit to 1 in the bus off state forcibly returns the state from the bus off state to the error active state. This bit is automatically cleared to 0. Setting this bit to 1 clears the TEC[7:0] and REC[7:0] bits in the RSCAN1CmSTS register to $00_H$ and also clears the BOSTS flag in the RSCAN1CmSTS register to 0 (not in bus off state). The other registers remain unchanged. No bus off recovery interrupt request is generated upon return from the bus off state in this case. Use this bit only when the BOM[1:0] bits in the RSCAN1CmCTR register are $00_B$ (ISO11898-1 compliant).

A delay of up to 1 CAN bit time occurs after the RTBO bit is set to 1 until the RSCAN module transitions to the error active state. Set this bit to 1 in channel communication mode.

#### CSLPR Bit

Setting this bit to 1 places the channel into channel stop mode.
Clearing this bit to 0 makes the channel exit channel stop mode.
This bit should not be modified in channel communication mode or channel halt mode.

#### CHMDC[1:0] Bits

These bits are used to select a channel mode (channel communication mode, channel reset mode, or channel halt mode). For details, see **Section 21.18.2, Channel Modes**. Setting the CSLPR bit to 1 in channel reset mode allows transition to channel stop mode. Do not set the CHMDC[1:0] bits to $11_B$. When the CAN module has automatically transitioned to channel halt mode based on the setting of the BOM[1:0] bits, the CHMDC[1:0] bits automatically become $10_B$.

## 21.16.4 RSCAN1CmSTS — Channel Status Register (m = 6)

**Access:** RSCAN1CmSTS register is a read-only register that can be read in 32-bit units.
RSCAN1CmSTSL, RSCAN1CmSTSH registers are read-only registers that can be read in 16-bit units.
RSCAN1CmSTSLL, RSCAN1CmSTSHL, RSCAN1CmSTSHH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1CmSTS: <RSCAN1_base> + $0008_H$ + ($10_H$ × i)

RSCAN1CmSTSL: <RSCAN1_base> + $0008_H$ + ($10_H$ × i),
RSCAN1CmSTSH: <RSCAN1_base> + $000A_H$ + ($10_H$ × i)

RSCAN1CmSTSLL: <RSCAN1_base> + $0008_H$ + ($10_H$ × i),
RSCAN1CmSTSHL: <RSCAN1_base> + $000A_H$ + ($10_H$ × i),
RSCAN1CmSTSHH: <RSCAN1_base> + $000B_H$ + ($10_H$ × i)

**Value after reset:** $0000\ 0005_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEC[7:0] | | | | | | | | REC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | COMSTS | RECSTS | TRMSTS | BOSTS | EPSTS | CSLPSTS | CHLTSTS | CRSTSTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.220 RSCAN1CmSTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | TEC[7:0] | The transmit error counter (TEC) can be read. |
| 23 to 16 | REC[7:0] | The receive error counter (REC) can be read. |
| 15 to 8 | Reserved | When read, the value after reset is returned. |
| 7 | COMSTS | Communication Status Flag<br>0: Communication is not ready.<br>1: Communication is ready. |
| 6 | RECSTS | Receive Status Flag<br>0: Bus idle, in transmission or bus off state<br>1: In reception |
| 5 | TRMSTS | Transmit Status Flag<br>0: Bus idle or in reception<br>1: In transmission or bus off state |
| 4 | BOSTS | Bus Off Status Flag<br>0: Not in bus off state<br>1: In bus off state |
| 3 | EPSTS | Error Passive Status Flag<br>0: Not in error passive state<br>1: In error passive state |
| 2 | CSLPSTS | Channel Stop Status Flag<br>0: Not in channel stop mode<br>1: In channel stop mode |
| 1 | CHLTSTS | Channel Halt Status Flag<br>0: Not in channel halt mode<br>1: In channel halt mode |
| 0 | CRSTSTS | Channel Reset Status Flag<br>0: Not in channel reset mode<br>1: In channel reset mode |

#### TEC[7:0] Bits

These bits contain the transmit error counter value. For transmit error counter increment/decrement conditions, see the CAN specification (ISO11898-1).

These bits are cleared to 0 in channel reset mode.

#### REC[7:0] Bits

These bits contain the receive error counter value. For receive error counter increment/decrement conditions, see the CAN specifications (ISO11898-1).

These bits are cleared to 0 in channel reset mode.

#### COMSTS Flag

This bit indicates that communication is ready.

This flag becomes 1 when the CAN module has detected 11 consecutive recessive bits after it has transitioned from channel reset mode or channel halt mode to channel communication mode. This flag is cleared to 0 in channel reset mode or channel halt mode.

#### RECSTS Flag

This flag is set to 1 when reception has started, and is cleared to 0 when the bus has become idle or transmission has started.

#### TRMSTS Flag

This flag is set to 1 when transmission has started, and is cleared to 0 when the bus has become idle or reception has started. This flag remains 1 in the bus off state.

#### BOSTS Flag

This flag is set to 1 when the bus off state (TEC[7:0] > 255) is entered. It is cleared to 0 when the CAN module has exited the bus off state.

#### EPSTS Flag

This flag is set to 1 when the RS-CAN module has entered the error passive state ((128 ≤ TEC[7:0] ≤ 255) or (128 ≤ REC[7:0])). It is cleared to 0 when the RS-CAN module has exited the error passive state or has entered channel reset mode.

#### CSLPSTS Flag

This flag is set to 1 when the CAN module has transitioned to channel stop mode, and is cleared to 0 when the CAN module has returned from channel stop mode.

#### CHLTSTS Flag

This flag is set to 1 when the CAN module has transitioned to channel halt mode, and is cleared to 0 when the CAN module has returned from channel halt mode.

#### CRSTSTS Flag

This flag is set to 1 when the CAN module has transitioned to channel reset mode, and is cleared to 0 when the CAN module has transitioned to channel communication mode or channel halt mode. This flag remains 1 when the CAN module transitions from channel reset mode to channel stop mode.

## 21.16.5 RSCAN1CmERFL — Channel Error Flag Register (m = 6)

**Access:** RSCAN1CmERFL register can be read or written in 32-bit units.
RSCAN1CmERFLL register can be read or written in 16-bit units.
RSCAN1CmERFLH register is a read-only register that can be read in 16-bit units.
RSCAN1CmERFLLL, RSCAN1CmERFLLH registers can be read or written in 8-bit units.
RSCAN1CmERFLHL, RSCAN1CmERFLHH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1CmERFL: <RSCAN1_base> + $000C_H$ + ($10_H$ × i)

RSCAN1CmERFLL: <RSCAN1_base> + $000C_H$ + ($10_H$ × i),
RSCAN1CmERFLH: <RSCAN1_base> + $000E_H$ + ($10_H$ × i)

RSCAN1CmERFLLL: <RSCAN1_base> + $000C_H$ + ($10_H$ × i),
RSCAN1CmERFLLH: <RSCAN1_base> + $000D_H$ + ($10_H$ × i),
RSCAN1CmERFLHL: <RSCAN1_base> + $000E_H$ + ($10_H$ × i),
RSCAN1CmERFLHH: <RSCAN1_base> + $000F_H$ + ($10_H$ × i)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | CRCREG[14:0] | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | ADERR | B0ERR | B1ERR | CERR | AERR | FERR | SERR | ALF | BLF | OVLF | BORF | BOEF | EPF | EWF | BEF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.221 RSCAN1CmERFL Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 30 to 16 | CRCREG[14:0] | CRC Calculation Data<br>A CRC value calculated based on the transmit message or receive message is indicated. |
| 15 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 14 | ADERR | ACK Delimiter Error Flag<br>0: No ACK delimiter error is detected.<br>1: ACK delimiter error is detected. |
| 13 | B0ERR | Dominant Bit Error Flag<br>0: No dominant bit error is detected.<br>1: Dominant bit error is detected. |
| 12 | B1ERR | Recessive Bit Error Flag<br>0: No recessive bit error is detected.<br>1: Recessive bit error is detected. |
| 11 | CERR | CRC Error Flag<br>0: No CRC error is detected.<br>1: CRC error is detected. |
| 10 | AERR | ACK Error Flag<br>0: No ACK error is detected.<br>1: ACK error is detected. |
| 9 | FERR | Form Error Flag<br>0: No form error is detected.<br>1: Form error is detected. |

Table 21.221 RSCAN1CmERFL Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 8 | SERR | Stuff Error Flag<br>0: No stuff error is detected.<br>1: Stuff error is detected. |
| 7 | ALF | Arbitration-lost Flag<br>0: No arbitration-lost is detected.<br>1: Arbitration-lost is detected. |
| 6 | BLF | Bus Lock Flag<br>0: No channel bus is detected.<br>1: Channel bus is detected. |
| 5 | OVLF | Overload Flag<br>0: No overload is detected.<br>1: Overload is detected. |
| 4 | BORF | Bus Off Recovery Flag<br>0: No bus off recovery is detected.<br>1: Bus off recovery is detected. |
| 3 | BOEF | Bus Off Entry Flag<br>0: No bus off entry is detected.<br>1: Bus off entry is detected. |
| 2 | EPF | Error Passive Flag<br>0: No error passive is detected.<br>1: Error passive is detected. |
| 1 | EWF | Error Warning Flag<br>0: No error warning is detected.<br>1: Error warning is detected. |
| 0 | BEF | Bus Error Flag<br>0: No channel bus error is detected.<br>1: Channel bus error is detected. |

See the CAN specification (ISO11898-1) for a description of error occurrence conditions. To clear each flag of this register, the program must write a 0 to the corresponding bit. These flags cannot be set to 1 by the program. If any of these flags is set to 0 at the same time that the program writes 0 to the flag, the flag is still set to 1. The channel reset mode transition clears all of these flags to 0.

If the ERRD bit in the RSCAN1CmCTR register is set to 0 (ie, only the flags for the first error event are displayed) and an error related to bits 14 to 8 of RSCAN1CmERFL is detected, the flag bits are only set by the error event if bits 14 to 8 were all 0 at the when time the error occurred.

#### CRCREG[14:0] Flags

When the CTME bit in the RSCAN1CmCTR register is set to 1 (communication test mode is enabled), the CRC value calculated based on the transmit or receive message can be read. When the CTME bit is set to 0 (communication test mode is disabled), these bits are always read as 0.

#### ADERR Flag

This flag is set to 1 when a form error has been detected in the ACK delimiter during transmission.

#### B0ERR Flag

This flag is set to 1 when a recessive bit has been detected though a dominant bit was transmitted.

#### B1ERR Flag

This flag is set to 1 when a dominant bit has been detected though a recessive bit was transmitted.

#### CERR Flag

This flag is set to 1 when a CRC error has been detected.

#### AERR Flag

This flag is set to 1 when an ACK error has been detected.

#### FERR Flag

This flag is set to 1 when a form error has been detected.

#### SERR Flag

This flag is set to 1 when a stuff error has been detected.

#### ALF Flag

This flag is set to 1 when an arbitration-lost has been detected.

#### BLF Flag

This flag is set to 1 when 32 consecutive dominant bits have been detected on the CAN bus in channel communication mode. After that, detection of a dominant lock is restarted when either of the following conditions is met.

- A recessive bit is detected after the BLF bit has been cleared from 1 to 0.
- The CAN module transitions to channel reset mode and returns to channel communication mode after the BLF bit has been cleared from 1 to 0.

#### OVLF Flag

This flag is set to 1 when the overload frame transmit condition has been detected when performing reception or transmission.

#### BORF Flag

This flag is set to 1 when 11 consecutive recessive bits have been detected 128 times and the CAN module returns from the bus off state. However, this flag is not set to 1 if the CAN module returns from the bus off state in any of the following ways before 11 consecutive recessive bits are detected 128 times.

- The CHMDC[1:0] bits in the RSCAN1CmCTR register are set to $01_B$ (channel reset mode).
- The RTBO bit in the RSCAN1CmCTR register is set to 1 (forcible return from the bus off state is made).
- The BOM[1:0] bits in the RSCAN1CmCTR register are set to $01_B$ (transition to channel halt mode at bus off entry).
- The CHMDC[1:0] bits in the RSCAN1CmCTR register are set to $10_B$ (channel halt mode) before 11 consecutive recessive bits are detected 128 times with the BOM[1:0] bits set to $11_B$ (transition to channel halt mode upon a request from the program during bus off).

### BOEF Flag

This flag is set to 1 when the bus off state is reached (TEC[7:0] value > 255). This flag is also set to 1 if the bus off state is reached when the BOM[1:0] bits in the RSCAN1CmCTR register (m = 6) set to $01_B$ (transition to channel halt mode at bus off entry).

### EPF Flag

This flag becomes 1 when the error passive state is reached (REC[7:0] or TEC[7:0] value > 127). This flag becomes 1 only when the REC[7:0] or TEC[7:0] value first exceeds 127. Therefore, if the program writes 0 to this flag while the value of REC[7:0] or TEC[7:0] remains over 127, this bit is not set to 1 until both REC [7:0] and TEC[7:0] values become 127 or less and then the REC[7:0] or TEC[7:0] value exceeds 127 again.

### EWF Flag

This flag is set to 1 only when the REC[7:0] or TEC[7:0] value first exceeds 95. Therefore, if the program writes 0 to this flag while the value of REC[7:0] or TEC[7:0] remains over 95, this bit is not set to 1 until both REC [7:0] and TEC[7:0] values become 95 or less and then the REC[7:0] or TEC[7:0] value exceeds 95 again.

### BEF Flag

This flag is set to 1 when any one of the ADERR, B0ERR, B1ERR, CERR, AERR, FERR, and SERR flags in the RSCAN1CmERFL register is set to 1.

**NOTE**

To clear the flag of this register to 0, use a store instruction to write “0” to the given flag and “1” to the other flags.

## 21.16.6 RSCAN1GCFG — Global Configuration Register

**Access:** RSCAN1GCFG register can be read or written in 32-bit units.
RSCAN1GCFGL, RSCAN1GCFGH registers can be read or written in 16-bit units.
RSCAN1GCFGLL, RSCAN1GCFGLH, RSCAN1GCFGHL, RSCAN1GCFGHH registers can be read or written in 8-bit units.

**Address:** RSCAN1GCFG: <RSCAN1_base> + $0084_H$

RSCAN1GCFGL: <RSCAN1_base> + $0084_H$,
RSCAN1GCFGH: <RSCAN1_base> + $0086_H$

RSCAN1GCFGLL: <RSCAN1_base> + $0084_H$,
RSCAN1GCFGLH: <RSCAN1_base> + $0085_H$,
RSCAN1GCFGHL: <RSCAN1_base> + $0086_H$,
RSCAN1GCFGHH: <RSCAN1_base> + $0087_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ITRCP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSBTCS[2:0] | | | TSSS | TSP[3:0] | | | | — | — | — | DCS | MME | DRE | DCE | TPRI |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 21.222 RSCAN1GCFG Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | ITRCP[15:0] | Interval Timer Prescaler Set<br>When these bits are set to M, the pclk is divided by M.<br>Setting $0000_H$ is prohibited when the interval timer is in use. |
| 15 to 13 | TSBTCS[2:0] | Timestamp Clock Source Select<br>b15 b14 b13<br>0 0 0: Unit channel number 0 bit time clock<br>0 0 1: Unit channel number 1 bit time clock<br>0 1 0: Channel 2 bit time clock<br>0 1 1: Channel 3 bit time clock<br>1 0 0: Channel 4 bit time clock<br>1 0 1: Channel 5 bit time clock<br>1 1 0: Setting prohibited<br>1 1 1: Setting prohibited |
| 12 | TSSS | Timestamp Source Select<br>0: pclk/2*1<br>1: Bit time clock |

Table 21.222 RSCAN1GCFG Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 to 8 | TSP[3:0] | Timestamp Clock Source Division<br>b11 b10 b9 b8<br>0 0 0 0: Not divided<br>0 0 0 1: Divided by 2<br>0 0 1 0: Divided by 4<br>0 0 1 1: Divided by 8<br>0 1 0 0: Divided by 16<br>0 1 0 1: Divided by 32<br>0 1 1 0: Divided by 64<br>0 1 1 1: Divided by 128<br>1 0 0 0: Divided by 256<br>1 0 0 1: Divided by 512<br>1 0 1 0: Divided by 1024<br>1 0 1 1: Divided by 2048<br>1 1 0 0: Divided by 4096<br>1 1 0 1: Divided by 8192<br>1 1 1 0: Divided by 16384<br>1 1 1 1: Divided by 32768 |
| 7 to 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 | DCS | CAN Clock Source Select*2<br>0: clkc<br>1: clk_xincan |
| 3 | MME | Mirror Function Enable<br>0: Mirror function is disabled.<br>1: Mirror function is enabled. |
| 2 | DRE | DLC Replacement Enable<br>0: DLC replacement is disabled.<br>1: DLC replacement is enabled. |
| 1 | DCE | DLC Check Enable<br>0: DLC check is disabled.<br>1: DLC check is enabled. |
| 0 | TPRI | Transmit Priority Select<br>0: ID priority<br>1: Transmit buffer number priority |

Note 1. When specifying pclk/2 as the timestamp counter count source, set bits TSBTCS[2:0] to $000_B$.

Note 2. For the CAN clock frequency settings, see **Table 21.208, Range of Operating Frequency Depending on the Transfer Rate and the Number of Channels in Use in the RH850/F1K**.

Modify the RSCAN1GCFG register only in global reset mode.

### ITRCP[15:0] Bits

These bits are used to set a clock source division value of the interval timer for FIFO buffers. See **Section 21.20.3.1, Interval Transmission Function**.

### TSBTCS[2:0] Bits

When the TSSS bit is 1, these bits are used to select the channel of the bit time clock that will be the clock source of the timestamp counter.

### TSSS Bit

This bit is used to select a clock source of the timestamp counter.

#### TSP[3:0] Bits

A clock obtained by dividing the clock source selected with the TSBTCS[2:0] bits and TSSS bit according to the TSP[3:0] bits is used as the timestamp counter count source.

#### DCS Bit

When this bit is set to 0, clkc is used as the clock source of the CAN clock (fCAN).

When this bit is set to 1, clk_xincan is used as the clock source of the CAN clock (fCAN).

For the CAN clock frequency settings, see **Table 21.208, Range of Operating Frequency Depending on the Transfer Rate and the Number of Channels in Use in the RH850/F1K**.

#### MME Bit

Setting this bit to 1 makes the mirror function available.

#### DRE Bit

When the DRE bit is set to 1, the DLC value of the receive rule is stored in the buffer instead of the DLC value of the received message after the DLC value has passed through the DLC filter. In this case, a value of $00_H$ is stored in each data byte beyond the DLC value of the receive rule.

The DLC replacement function is only available when the DCE bit is set to 1 (DLC check is enabled).

#### DCE Bit

Setting this bit to 1 makes the DLC check function available. When disabling the DLC check function, set the GAFLDLC[3:0] bits in the RSCAN1GAFLP0j register to $0000_B$ before clearing the DCE bit in the RSCAN1GCFG register to 0.

#### TPRI Bit

This bit is used to set the transmit priority.

When this bit is set to 0, ID priority is selected and the transmit priority complies with the CAN bus arbitration rule (ISO11898-1 specifications). When this bit is set to 1, transmit buffer number priority is selected and the lowest transmit buffer number of those has the highest priority.

While the transmit queue is in use, this bit should be set to 0.

## 21.16.7 RSCAN1GCTR — Global Control Register

**Access:** RSCAN1GCTR register can be read or written in 32-bit units.
RSCAN1GCTRL, RSCAN1GCTRH registers can be read or written in 16-bit units.
RSCAN1GCTRLL, RSCAN1GCTRLH, RSCAN1GCTRHL registers can be read or written in 8-bit units.

**Address:** RSCAN1GCTR: <RSCAN1_base> + $0088_H$

RSCAN1GCTRL: <RSCAN1_base> + $0088_H$,
RSCAN1GCTRH: <RSCAN1_base> + $008A_H$

RSCAN1GCTRLL: <RSCAN1_base> + $0088_H$,
RSCAN1GCTRLH: <RSCAN1_base> + $0089_H$,
RSCAN1GCTRHL: <RSCAN1_base> + $008A_H$

**Value after reset:** $0000\ 0005_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TSRST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | THLEIE | MEIE | DEIE | — | — | — | — | — | GSLPR | GMDC[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

**Table 21.223 RSCAN1GCTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 17 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 16 | TSRST | Timestamp Counter Reset<br>Setting the TSRST bit to 1 resets the timestamp counter. This bit is always read as 0. |
| 15 to 11 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 10 | THLEIE | Transmit History Buffer Overflow Interrupt Enable<br>0: Transmit history buffer overflow interrupt is disabled.<br>1: Transmit history buffer overflow interrupt is enabled. |
| 9 | MEIE | FIFO Message Lost Interrupt Enable<br>0: FIFO message lost interrupt is disabled.<br>1: FIFO message lost interrupt is enabled. |
| 8 | DEIE | DLC Error Interrupt Enable<br>0: DLC error interrupt is disabled.<br>1: DLC error interrupt is enabled. |
| 7 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 | GSLPR | Global Stop Mode<br>0: Other than global stop mode<br>1: Global stop mode |
| 1, 0 | GMDC[1:0] | Global Mode Select<br>b1 b0<br>0 0: Global operating mode<br>0 1: Global reset mode<br>1 0: Global test mode<br>1 1: Setting prohibited |

#### TSRST Bit

This bit is used to reset the timestamp counter. When this bit is set to 1, the RSCAN1GTSC register is cleared to $0000_H$.

#### THLEIE Bit

When the THLEIE bit is set to 1 and the THLES flag in the RSCAN1GERFL register is set to 1, an interrupt request is generated. Modify this bit only in global reset mode.

#### MEIE Bit

When the MEIE bit is set to 1 and the MES flag in the RSCAN1GERFL register is set to 1, an interrupt request is generated. Modify this bit only in global reset mode.

#### DEIE Bit

When the DEIE bit is set to 1 and the DEF flag in the RSCAN1GERFL register is set to 1, an interrupt request is generated. Modify this bit only in global reset mode.

#### GSLPR Bit

Setting this bit to 1 places the RSCAN module into global stop mode.
Clearing this bit to 0 makes the RSCAN module leave from global stop mode.
This bit should not be modified in global operating mode or global test mode.

#### GMDC[1:0] Bits

These bits are used to select the mode of entire RS-CAN module (global operating mode, global reset mode, or global test mode). For details, see **Section 21.18.1, Global Modes**. Setting the GSLPR bit to 1 when in global reset mode places the RS-CAN module into global stop mode.

## 21.16.8 RSCAN1GSTS — Global Status Register

**Access:** RSCAN1GSTS register is a read-only register that can be read in 32-bit units.
RSCAN1GSTSL register is a read-only register that can be read in 16-bit units.
RSCAN1GSTSLL register is a read-only register that can be read in 8-bit units.

**Address:** RSCAN1GSTS: <RSCAN1_base> + $008C_H$

RSCAN1GSTSL: <RSCAN1_base> + $008C_H$

RSCAN1GSTSLL: <RSCAN1_base> + $008C_H$

**Value after reset:** $0000\ 000D_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | GRAMINIT | GSLPSTS | GHLTSTS | GRSTSTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.224 RSCAN1GSTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is returned. |
| 3 | GRAMINIT | CAN RAM Initialization Status Flag<br>0: CAN RAM initialization is completed.<br>1: CAN RAM initialization is ongoing. |
| 2 | GSLPSTS | Global Stop Status Flag<br>0: Not in global stop mode<br>1: In global stop mode |
| 1 | GHLTSTS | Global Test Status Flag<br>0: Not in global test mode<br>1: In global test mode |
| 0 | GRSTSTS | Global Reset Status Flag<br>0: Not in global reset mode<br>1: In global reset mode |

### GRAMINIT Flag

This flag indicates the initialization status of the CAN RAM.

This flag is set to 1 after the MCU has been reset, and is cleared to 0 when CAN RAM initialisation is completed.

### GSLPSTS Flag

This flag is set to 1 when the CAN module has transitioned to global stop mode, and is cleared to 0 when the CAN module has returned from global stop mode.

### GHLTSTS Flag

This flag is set to 1 when the CAN module has transitioned to global test mode, and is cleared to 0 when the CAN module has exited global test mode.

**GRSTSTS Flag**

This flag is set to 1 when the CAN module has transitioned to global reset mode, and is cleared to 0 when the CAN module has exited global reset mode. This flag remains 1 even when the CAN module has transitioned from global reset mode to global stop mode.

## 21.16.9 RSCAN1GERFL — Global Error Flag Register

**Access:** RSCAN1GERFL register can be read or written in 32-bit units.
RSCAN1GERFLL register can be read or written in 16-bit units.
RSCAN1GERFLLL register can be read or written in 8-bit units.

**Address:** RSCAN1GERFL: <RSCAN1_base> + $0090_H$

RSCAN1GERFLL: <RSCAN1_base> + $0090_H$

RSCAN1GERFLLL: <RSCAN1_base> + $0090_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | THLES | MES | DEF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W*1 |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.225 RSCAN1GERFL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 14 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 13 to 8 | Reserved | When read, the undefined value is returned. When writing, write the value after reset. |
| 7, 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5, 4 | Reserved | When read, the undefined value is returned. When writing, write the value after reset. |
| 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 | THLES | Transmit History Buffer Overflow Status Flag<br>0: No transmit history buffer overflow has occurred.<br>1: A transmit history buffer overflow has occurred. |
| 1 | MES | FIFO Message Lost Status Flag<br>0: No FIFO message lost error has occurred.<br>1: A FIFO message lost error has occurred. |
| 0 | DEF | DLC Error Flag<br>0: No DLC error has occurred.<br>1: A DLC error has occurred. |

All flags in the RSCAN1GERFL register are cleared to 0 in global reset mode.

### THLES Flag

The THLES flag is set to 1 when any one of the THLELT flags in the RSCAN1THLSTSm register (m = 6) is set to 1.

This flag is cleared to 0 when the THLELT flags of all channels are set to 0.

### MES Flag

The MES flag is set to 1 when any one of the RFMLT flags in the RSCAN1RFSTSx register (x = 0 to 7) or the CFMLT flags in the RSCAN1CFSTSk register (k = 0 to 2) is set to 1.

This flag is cleared to 0 when all RFMLT flags and CFMLT flags are set to 0.

### DEF Flag

The DEF flag is set to 1 when an error has been detected during the DLC check. The program can clear this flag by writing 0 to this bit.

**NOTE**

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared. When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

## 21.16.10 RSCAN1GTINTSTS0 — Global TX Interrupt Status Register 0

**Access:** RSCAN1GTINTSTS0 register is a read-only register that can be read in 32-bit units.
RSCAN1GTINTSTS0L register is a read-only register that can be read in 16-bit units.
RSCAN1GTINTSTS0LL register is a read-only register that can be read in 8-bit units.

**Address:** RSCAN1GTINTSTS0: <RSCAN1_base> + $0460_H$

RSCAN1GTINTSTS0L: <RSCAN1_base> + $0460_H$

RSCAN1GTINTSTS0LL: <RSCAN1_base> + $0460_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | THIF0 | CFTIF0 | TQIF0 | TAIF0 | TSIF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R*1 | R*1 | R*1 | R*1 | R*1 |

Note 1. This bit is automatically cleared in the global reset or channel reset mode.

**Table 21.226 RSCAN1GTINTSTS0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | When read, the value after reset is returned. |
| 4 | THIF0 | Unit Channel Number 0 Transmit History Interrupt Status Flag<br>0: Transmit history interrupt is not requested.<br>1: Transmit history interrupt is requested. |
| 3 | CFTIF0 | Unit Channel Number 0 Transmit/receive FIFO Transmit Interrupt Status Flag<br>0: Transmit/receive FIFO transmit interrupt is not requested.<br>1: Transmit/receive FIFO transmit interrupt is requested. |
| 2 | TQIF0 | Unit Channel Number 0 Transmit Queue Interrupt Status Flag<br>0: Transmit queue interrupt is not requested.<br>1: Transmit queue interrupt is requested. |
| 1 | TAIF0 | Unit Channel Number 0 Transmit Buffer Abort Interrupt Status Flag<br>0: Transmit buffer abort interrupt is not requested.<br>1: Transmit buffer abort interrupt is requested. |
| 0 | TSIF0 | Unit Channel Number 0 Transmit Buffer Interrupt Status Flag<br>0: Transmit buffer transmit complete interrupt is not requested.<br>1: Transmit buffer transmit complete interrupt is requested. |

### TSIFi Bits

The TSIFi bit is set to 1 when the TMIEp bit in the RSCAN1TMIECy register is set to 1 (transmit buffer interrupt enabled) and the TMTRF[1:0] flags in the RSCAN1TMSTSp register are set to $10_B$ (transmit completed without abort request) or $11_B$ (transmit completed with abort request).

When the TMTRF[1:0] flags are cleared to $00_B$ under the condition that the TSIFi bit can be set to 1, this flag is cleared to 0. In addition, clearing the TMIEp bit to 0 also clears this flag to 0.

#### TAIFi Bits

The TAIFi bit is set to 1 when the TAIE bit in the RSCAN1CmCTR register is 1 (transmit abort interrupt enabled) and the TMTRF[1:0] flags in the RSCAN1TMSTSp register are set to $01_B$ (transmit abort completed).

This flag is cleared to 0 when the TMTRF[1:0] flags are cleared to $00_B$ after the transmit abort is completed.

#### TQIFi Bits

When the TXQIE bit in the RSCAN1TXQCCm register is set to 1 (transmit queue interrupt enabled) and the TXQIF bit in the RSCAN1TXQSTSm register is set to 1 (transmit queue interrupt request), the TQIFi bit is set to 1.

When the TXQIF bit (transmit queue interrupt request) in the RSCAN1TXQSTSm register is cleared to 0, this bit is cleared to 0. This flag is also cleared to 0 when the TXQIE bit is cleared to 0.

#### CFTIFi Bits

When the CFTXIE bit in the RSCAN1CFCCk register is set to 1 (transmit/receive FIFO buffer transmit interrupt enabled) and the CFTXIF bit in the RSCAN1CFSTSk register is set to 1 (transmit/receive FIFO transmit interrupt request), the CFTIFi bit is set to 1.

When the CFTIFi bit is cleared to 0 under the conditions that the CFTIFi bit can be set to 1, this bit is cleared to 0. This flag is also cleared to 0 when the CFTXIE bit is cleared to 0.

#### THIFi Bits

When the THLIE bit in the RSCAN1THLCCm register is set to 1 (transmit history interrupt enabled) and the THLIF bit in the RSCAN1THLSTSm register is set to 1 (transmit history interrupt request), the THIFi bit is set to 1.

When the THLIF bit in the RSCAN1THLSTSm register is cleared to 0, this bit is cleared to 0. This flag is also cleared to 0 when the THLIE bit is cleared to 0.

## 21.16.11 RSCAN1GTSC — Global Timestamp Counter Register

**Access:** RSCAN1GTSC register is a read-only register that can be read in 32-bit units.
RSCAN1GTSCL register is a read-only register that can be read in 16-bit units.

**Address:** RSCAN1GTSC: <RSCAN1_base> + $0094_H$

RSCAN1GTSCL: <RSCAN1_base> + $0094_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.227 RSCAN1GTSC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. |
| 15 to 0 | TS[15:0] | Timestamp Value<br>The timestamp counter value can be read.<br>Counter Value: $0000_H$ to $FFFF_H$ |

### TS[15:0] Bits

When the TS[15:0] bits are read, the read value shows the timestamp counter (16-bit free-running counter) value at that time. When the SOF is detected, the TS[15:0] value is captured and later stored in the receive buffer or the FIFO buffer. The timestamp counter is initialized in global reset mode.

The timestamp counter starts and stops counting differently, depending on the count source.

- When the TSSS bit in the RSCAN1GCFG register is 0 (pclk):
  The timestamp counter starts counting when the RSCAN module has transitioned to global operating mode.
  This counter stops counting when the RSCAN module has transitioned to global stop mode or global test mode.
- When the TSSS bit is 1 (CANm bit time clock):
  The timestamp counter starts counting when the corresponding channel has transitioned to channel communication mode.
  This counter stops counting when the corresponding channel has transitioned to channel reset mode or channel halt mode.

## 21.16.12 RSCAN1GAFLECTR — Receive Rule Entry Control Register

**Access:** RSCAN1GAFLECTR register can be read or written in 32-bit units.
RSCAN1GAFLECTRL register can be read or written in 16-bit units.
RSCAN1GAFLECTRLL, RSCAN1GAFLECTRLH registers can be read or written in 8-bit units.

**Address:** RSCAN1GAFLECTR: <RSCAN1_base> + $0098_H$

RSCAN1GAFLECTRL: <RSCAN1_base> + $0098_H$

RSCAN1GAFLECTRLL: <RSCAN1_base> + $0098_H$,
RSCAN1GAFLECTRLH: <RSCAN1_base> + $0099_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | AFLDAE | — | — | — | AFLPN[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 21.228 RSCAN1GAFLECTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 8 | AFLDAE | Receive Rule Table Write Enable<br>0: Receive rule table write is disabled.<br>1: Receive rule table write is enabled. |
| 7 to 5 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 4 to 0 | AFLPN[4:0] | Receive Rule Table Page Number Configuration<br>A page number can be selected from a range of page 0 ($00000_B$) to page 23 ($10111_B$). |

### AFLDAE Bit

Setting this bit to 0 disables the write to the receive rule table. After writes to the receive rule table are completed, set this bit to 0 to disable the write to the table. The receive rule table can be read regardless of the value of this bit.

Set the AFLDAE bit to 1 only in global reset mode.

### AFLPN[4:0] Bits

These bits are used to set the page number of the receive rule table. Sixteen receive rules can be set per page.

Set these bits to a value within the range of $00000_B$ to $10111_B$.

## 21.16.13 RSCAN1GAFLCFG0 — Receive Rule Configuration Register 0

**Access:** RSCAN1GAFLCFG0 register can be read or written in 32-bit units.
RSCAN1GAFLCFG0H register can be read or written in 16-bit units.
RSCAN1GAFLCFG0HH register can be read or written in 8-bit units.

**Address:** RSCAN1GAFLCFG0: <RSCAN1_base> + $009C_H$

RSCAN1GAFLCFG0H: <RSCAN1_base> + $009E_H$

RSCAN1GAFLCFG0HH: <RSCAN1_base> + $009F_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RNC0[7:0] | | | | | | | | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.229 RSCAN1GAFLCFG0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RNC0[7:0] | Number of Rules for Unit Channel Number 0<br>Set the number of receive rules exclusively used for Unit Channel Number 0. |
| 23 to 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Modify the RSCAN1GAFLCFG0 register only in global reset mode.

Up to 64 × (number of channels) rules can be registered in the receive rule table as the entire unit. The number of receive rules for each channel of the units should meet the following conditions.

- The maximum number of rules per channel is 128.
- The total of the number of rules allocated to each channel is not larger than the number of rules that can be registered in the entire unit.

### RNC0[7:0] Bits

These bits are used to set the number of rules to be registered in the receive rule table for unit channel number 0.

Set these bits to a value within the range of $00_H$ to $80_H$.

## 21.16.14 RSCAN1GAFLIDj — Receive Rule ID Register (j = 0 to 15)

**Access:** RSCAN1GAFLIDj register can be read or written in 32-bit units.
RSCAN1GAFLIDjL, RSCAN1GAFLIDjH registers can be read or written in 16-bit units.
RSCAN1GAFLIDjLL, RSCAN1GAFLIDjLH, RSCAN1GAFLIDjHL, RSCAN1GAFLIDjHH registers can be read or written in 8-bit units.

**Address:** RSCAN1GAFLIDj: <RSCAN1_base> + $0500_H$ + ($10_H$ × j)

RSCAN1GAFLIDjL: <RSCAN1_base> + $0500_H$ + ($10_H$ × j),
RSCAN1GAFLIDjH: <RSCAN1_base> + $0502_H$ + ($10_H$ × j)

RSCAN1GAFLIDjLL: <RSCAN1_base> + $0500_H$ + ($10_H$ × j),
RSCAN1GAFLIDjLH: <RSCAN1_base> + $0501_H$ + ($10_H$ × j),
RSCAN1GAFLIDjHL: <RSCAN1_base> + $0502_H$ + ($10_H$ × j),
RSCAN1GAFLIDjHH: <RSCAN1_base> + $0503_H$ + ($10_H$ × j)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLIDE | GAFLRTR | GAFLLB | GAFLID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.230 RSCAN1GAFLIDj Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | GAFLIDE | IDE Select<br>0: Standard ID<br>1: Extended ID |
| 30 | GAFLRTR | RTR Select<br>0: Data frame<br>1: Remote frame |
| 29 | GAFLLB | Receive Rule Target Message Select<br>0: When a message transmitted from another CAN node is received<br>1: When the own transmitted message is received |
| 28 to 0 | GAFLID[28:0] | ID Set<br>Set the ID of the receive rule.<br>For the standard ID, set the ID in bits b10 to b0 and set bits b28 to b11 to 0. |

Modify the RSCAN1GAFLIDj register when the AFLDAE bit in the RSCAN1GAFLECTR register is set to 1 (receive rule table write is enabled) in global reset mode.

### GAFLIDE Bit

This bit is used to select the ID format (standard ID or extended ID) of the receive rule. This bit is compared with the IDE bit in the received message during the acceptance filter processing.

### GAFLRTR Bit

This bit is used to select the frame format (data frame or remote frame) of the receive rule. This bit is compared with the RTR bit in the received message during the acceptance filter processing.

#### GAFLLB Bit

When this bit is set to 0, data processing using the receive rule is performed when receiving messages transmitted from another CAN node.

When this bit is set to 1 when the mirror function is used, data processing using the receive rule is performed when the CAN node is receiving its own transmitted messages.

#### GAFLID[28:0] Bits

These bits are used to set the ID field of the receive rule. The ID value set by these bits is compared with the ID of the received message during the acceptance filter processing.

## 21.16.15 RSCAN1GAFLMj — Receive Rule Mask Register (j = 0 to 15)

**Access:** RSCAN1GAFLMj register can be read or written in 32-bit units.
RSCAN1GAFLMjL, RSCAN1GAFLMjH registers can be read or written in 16-bit units.
RSCAN1GAFLMjLL, RSCAN1GAFLMjLH, RSCAN1GAFLMjHL, RSCAN1GAFLMjHH registers can be read or written in 8-bit units.

**Address:** RSCAN1GAFLMj: <RSCAN1_base> + $0504_H$ + ($10_H$ × j)

RSCAN1GAFLMjL: <RSCAN1_base> + $0504_H$ + ($10_H$ × j),
RSCAN1GAFLMjH: <RSCAN1_base> + $0506_H$ + ($10_H$ × j)

RSCAN1GAFLMjLL: <RSCAN1_base> + $0504_H$ + ($10_H$ × j),
RSCAN1GAFLMjLH: <RSCAN1_base> + $0505_H$ + ($10_H$ × j),
RSCAN1GAFLMjHL: <RSCAN1_base> + $0506_H$ + ($10_H$ × j),
RSCAN1GAFLMjHH: <RSCAN1_base> + $0507_H$ + ($10_H$ × j)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLIDEM | GAFLRTRM | — | GAFLIDM[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLIDM[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.231 RSCAN1GAFLMj Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | GAFLIDEM | IDE Mask<br>0: The IDE bit is not compared.<br>1: The IDE bit is compared. |
| 30 | GAFLRTRM | RTR Mask<br>0: The RTR bit is not compared.<br>1: The RTR bit is compared |
| 29 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 28 to 0 | GAFLIDM[28:0] | ID Mask<br>0: The corresponding ID bit is not compared.<br>1: The corresponding ID bit is compared. |

Modify the RSCAN1GAFLMj register when the AFLDAE bit in the RSCAN1GAFLECTR register is set to 1 (receive rule table write is enabled) in global reset mode.

### GAFLIDEM Bit

When this bit is set to 1, filter processing is performed only for messages of the ID format specified by the GAFLIDE bit in the RSCAN1GAFLIDj register.

When this bit is cleared to 0, the IDs of all the receive messages and the specified IDs are regarded as matched. To set the GAFLIDEM bit to 0, set the GAFLIDM[28:0] bits to all 0 at the same time.

### GAFLRTRM Bit

This bit is used to mask the RTR bit of the receive rule.

### GAFLIDM[28:0] Bits

These bits are used to mask the corresponding ID bit of the receive rule.

## 21.16.16 RSCAN1GAFLP0j — Receive Rule Pointer 0 Register (j = 0 to 15)

**Access:** RSCAN1GAFLP0j register can be read or written in 32-bit units.
RSCAN1GAFLP0jL, RSCAN1GAFLP0jH registers can be read or written in 16-bit units.
RSCAN1GAFLP0jLH, RSCAN1GAFLP0jHL, RSCAN1GAFLP0jHH registers can be read or written in 8-bit units.

**Address:** RSCAN1GAFLP0j: <RSCAN1_base> + $0508_H$ + ($10_H$ × j)

RSCAN1GAFLP0jL: <RSCAN1_base> + $0508_H$ + ($10_H$ × j),
RSCAN1GAFLP0jH: <RSCAN1_base> + $050A_H$ + ($10_H$ × j)

RSCAN1GAFLP0jLH: <RSCAN1_base> + $0509_H$ + ($10_H$ × j),
RSCAN1GAFLP0jHL: <RSCAN1_base> + $050A_H$ + ($10_H$ × j),
RSCAN1GAFLP0jHH: <RSCAN1_base> + $050B_H$ + ($10_H$ × j)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLDLC[3:0] | | | | GAFLPTR[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAFLRMV | GAFLRMDP[6:0] | | | | | | | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

**Table 21.232 RSCAN1GAFLP0j Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | GAFLDLC[3:0] | Receive Rule DLC<br>b31 b30 b29 b28<br>0 0 0 0: DLC check is disabled.<br>0 0 0 1: 1 data byte<br>0 0 1 0: 2 data bytes<br>0 0 1 1: 3 data bytes<br>0 1 0 0: 4 data bytes<br>0 1 0 1: 5 data bytes<br>0 1 1 0: 6 data bytes<br>0 1 1 1: 7 data bytes<br>1 X X X: 8 data bytes |
| 27 to 16 | GAFLPTR[11:0] | Receive Rule Label<br>Set the 12-bit label information. |
| 15 | GAFLRMV | Receive Buffer Enable<br>0: No receive buffer is used.<br>1: A receive buffer is used. |
| 14 to 8 | GAFLRMDP[6:0] | Receive Buffer Number Select<br>Set the receive buffer number to store receive messages. |
| 7 to 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Modify the RSCAN1GAFLP0j register when the AFLDAE bit in the RSCAN1GAFLECTR register is set to 1 (receive rule table write is enabled) in global reset mode.

### GAFLDLC[3:0] Bits

These bits are used to set the minimum data length necessary for receiving messages. If the data length of a message that is being filtered is equal to or larger than the value set by the GAFLDLC[3:0] bits, the message passes the DLC check. Setting these bits to $0000_B$ disables the DLC check function allowing messages with any data length to pass the DLC check.

#### GAFLPTR[11:0] Bits

These bits are used to set a 12-bit label to be attached to messages that have passed through the filter. A label is attached when a message is stored in the receive buffer or the FIFO buffer.

#### GAFLRMV Bit

When this bit is set to 1, receive messages that have passed through the filter are stored in the receive buffer selected by the GAFLRMDP[6:0] bits.

#### GAFLRMDP[6:0] Bits

These bits are used to select the number of the receive buffer that stores receive messages that have passed through the filter when the GAFLRMV bit is set to 1. Set these bits to a value smaller than the value set by the NRXMB[7:0] bits in the RSCAN1RMNB register.

## 21.16.17 RSCAN1GAFLP1j — Receive Rule Pointer 1 Register (j = 0 to 15)

**Access:** RSCAN1GAFLP1j register can be read or written in 32-bit units.
RSCAN1GAFLP1jL register can be read or written in 16-bit units.
RSCAN1GAFLP1jLL, RSCAN1GAFLP1jLH registers can be read or written in 8-bit units.

**Address:** RSCAN1GAFLP1j: <RSCAN1_base> + $050C_H$ + ($10_H$ × j)

RSCAN1GAFLP1jL: <RSCAN1_base> + $050C_H$ + ($10_H$ × j)

RSCAN1GAFLP1jLL: <RSCAN1_base> + $050C_H$ + ($10_H$ × j),
RSCAN1GAFLP1jLH: <RSCAN1_base> + $050D_H$ + ($10_H$ × j)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | GAFLFDP[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.233 RSCAN1GAFLP1j Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 10 to 8 | GAFLFDP[10:8] | Transmit/Receive FIFO Buffer k Select<br>(Bit position –8 = target transmit/receive FIFO buffer number k)<br>0: Transmit/receive FIFO buffer is not selected.<br>1: Transmit/receive FIFO buffer is selected. |
| 7 to 0 | GAFLFDP[7:0] | Receive FIFO Buffer x Select<br>(Bit position = target receive FIFO buffer number x)<br>0: Receive FIFO buffer is not selected.<br>1: Receive FIFO buffer is selected. |

Modify the RSCAN1GAFLP1j register when the AFLDAE bit in the RSCAN1GAFLECTR register is set to 1 (receive rule table write is enabled) in global reset mode.

### GAFLFDP [10:0] Bits

These bits are used to specify FIFO buffers that store receive messages that have passed through the filter. Up to eight FIFO buffers can be selected. However, when the GAFLRMV bit in the RSCAN1GAFLP0j register is set to 1 (a message is stored in the receive buffer), up to seven FIFO buffers can be selected. Only receive FIFO buffers and the transmit/receive FIFO buffer for which the CFM[1:0] bits in the RSCAN1CFCCk register are set to $00_B$ (receive mode) can be selected.

## 21.16.18 RSCAN1RMNB — Receive Buffer Number Register

**Access:** RSCAN1RMNB register can be read or written in 32-bit units.
RSCAN1RMNBL register can be read or written in 16-bit units.
RSCAN1RMNBLL register can be read or written in 8-bit units.

**Address:** RSCAN1RMNB: <RSCAN1_base> + $00A4_H$

RSCAN1RMNBL: <RSCAN1_base> + $00A4_H$

RSCAN1RMNBLL: <RSCAN1_base> + $00A4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | NRXMB[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.234 RSCAN1RMNB Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 7 to 0 | NRXMB[7:0] | Receive Buffer Number Configuration<br>Set the number of receive buffers.<br>Set a value of 0 to 16. |

Modify the RSCAN1RMNB register only in global reset mode.

### NRXMB[7:0] Bits

These bits are used to set the total number of receive buffers of the RS-CAN module. The maximum value is 16 × (number of channels).

Setting all of these bits to 0 makes receive buffers unavailable.

## 21.16.19 RSCAN1RMNDy — Receive Buffer New Data Register (y = 0)

**Access:** RSCAN1RMNDy register can be read or written in 32-bit units.
RSCAN1RMNDyL register can be read or written in 16-bit units.
RSCAN1RMNDyLL, RSCAN1RMNDyLH registers can be read or written in 8-bit units.

**Address:** RSCAN1RMNDy: <RSCAN1_base> + $00A8_H$ + ($04_H$ × y)

RSCAN1RMNDyL: <RSCAN1_base> + $00A8_H$ + ($04_H$ × y)

RSCAN1RMNDyLL: <RSCAN1_base> + $00A8_H$ + ($04_H$ × y),
RSCAN1RMNDyLH: <RSCAN1_base> + $00A9_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMNSq (q = y × 32 + 15 to y × 32 + 0 (y = 0)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.235 RSCAN1RMNDy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 15 to 0 | RMNSq | Receive Buffer Receive Complete Flag q (q = y × 32 + 15 to y × 32 + 0)<br>0: There is no new message in receive buffer q.<br>1: There is a new message in receive buffer q. |

Write 0 to the RSCAN1RMNDy register in global operating mode or global test mode.

### RMNSq Flags (q = 0 to i × 16 + 15)

Each RMNS flag is set to 1 when the processing for storing a message in the corresponding receive buffer starts.

To clear a flag to 0, the program must write 0 to the flag. Use a store instruction to write “0” to the flag and “1” to other flags. These bits cannot be set to 0 while a message is being stored. It takes ten clock cycles of pclk to store a message.

These flags are cleared to 0 in global reset mode.

## 21.16.20 RSCAN1RMIDq — Receive Buffer ID Register (q = 0 to 15)

**Access:** RSCAN1RMIDq register is a read-only register that can be read in 32-bit units.
RSCAN1RMIDqL, RSCAN1RMIDqH registers are read-only registers that can be read in 16-bit units.
RSCAN1RMIDqLL, RSCAN1RMIDqLH, RSCAN1RMIDqHL, RSCAN1RMIDqHH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1RMIDq: <RSCAN1_base> + $0600_H$ + ($10_H$ × q)

RSCAN1RMIDqL: <RSCAN1_base> + $0600_H$ + ($10_H$ × q),
RSCAN1RMIDqH: <RSCAN1_base> + $0602_H$ + ($10_H$ × q)

RSCAN1RMIDqLL: <RSCAN1_base> + $0600_H$ + ($10_H$ × q),
RSCAN1RMIDqLH: <RSCAN1_base> + $0601_H$ + ($10_H$ × q),
RSCAN1RMIDqHL: <RSCAN1_base> + $0602_H$ + ($10_H$ × q),
RSCAN1RMIDqHH: <RSCAN1_base> + $0603_H$ + ($10_H$ × q)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMIDE | RMRTR | — | RMID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.236 RSCAN1RMIDq Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | RMIDE | Receive Buffer IDE<br>0: Standard ID<br>1: Extended ID |
| 30 | RMRTR | Receive Buffer RTR<br>0: Data frame<br>1: Remote frame |
| 29 | Reserved | When read, the value after reset is returned. |
| 28 to 0 | RMID[28:0] | Receive Buffer ID Data<br>These bits contain the standard ID or extended ID of the received message.<br>Read bits b10 to b0 for standard ID. Bits b28 to b11 are read as 0. |

### RMIDE Bit

This bit indicates the ID format (standard ID or extended ID) of the message stored in the receive buffer.

### RMRTR Bit

This bit indicates the frame format (data frame or remote frame) of the message stored in the receive buffer.

### RMID[28:0] Bits

These bits contain the ID of the message stored in the receive buffer.

## 21.16.21 RSCAN1RMPTRq — Receive Buffer Pointer Register (q = 0 to 15)

**Access:** RSCAN1RMPTRq register is a read-only register that can be read in 32-bit units.
RSCAN1RMPTRqL, RSCAN1RMPTRqH registers are read-only registers that can be read in 16-bit units.
RSCAN1RMPTRqLL, RSCAN1RMPTRqLH, RSCAN1RMPTRqHL, RSCAN1RMPTRqHH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1RMPTRq: <RSCAN1_base> + $0604_H$ + ($10_H$ × q)

RSCAN1RMPTRqL: <RSCAN1_base> + $0604_H$ + ($10_H$ × q),
RSCAN1RMPTRqH: <RSCAN1_base> + $0606_H$ + ($10_H$ × q)

RSCAN1RMPTRqLL: <RSCAN1_base> + $0604_H$ + ($10_H$ × q),
RSCAN1RMPTRqLH: <RSCAN1_base> + $0605_H$ + ($10_H$ × q),
RSCAN1RMPTRqHL: <RSCAN1_base> + $0606_H$ + ($10_H$ × q),
RSCAN1RMPTRqHH: <RSCAN1_base> + $0607_H$ + ($10_H$ × q)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMDLC[3:0] | | | | RMPTR[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMTS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.237 RSCAN1RMPTRq Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | RMDLC[3:0] | Receive Buffer DLC Data<br>b31 b30 b29 b28<br>0 0 0 0: No data byte<br>0 0 0 1: 1 data byte<br>0 0 1 0: 2 data bytes<br>0 0 1 1: 3 data bytes<br>0 1 0 0: 4 data bytes<br>0 1 0 1: 5 data bytes<br>0 1 1 0: 6 data bytes<br>0 1 1 1: 7 data bytes<br>1 X X X: 8 data bytes |
| 27 to 16 | RMPTR[11:0] | Receive Buffer Label Data<br>Label information of the received message. |
| 15 to 0 | RMTS[15:0] | Receive Buffer Timestamp Data<br>Timestamp value of the received message. |

### RMDLC[3:0] Bits

These bits indicate the data length of the message stored in the receive buffer.

### RMPTR[11:0] Bits

These bits indicate the label information of the message stored in the receive buffer.

### RMTS[15:0] Bits

These bits indicate the timestamp value of the message stored in the receive buffer.

## 21.16.22 RSCAN1RMDF0q — Receive Buffer Data Field 0 Register (q = 0 to 15)

**Access:** RSCAN1RMDF0q register is a read-only register that can be read in 32-bit units.
RSCAN1RMDF0qL, RSCAN1RMDF0qH registers are read-only registers that can be read in 16-bit units.
RSCAN1RMDF0qLL, RSCAN1RMDF0qLH, RSCAN1RMDF0qHL, RSCAN1RMDF0qHH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1RMDF0q: <RSCAN1_base> + $0608_H$ + ($10_H$ × q)

RSCAN1RMDF0qL: <RSCAN1_base> + $0608_H$ + ($10_H$ × q),
RSCAN1RMDF0qH: <RSCAN1_base> + $060A_H$ + ($10_H$ × q)

RSCAN1RMDF0qLL: <RSCAN1_base> + $0608_H$ + ($10_H$ × q),
RSCAN1RMDF0qLH: <RSCAN1_base> + $0609_H$ + ($10_H$ × q),
RSCAN1RMDF0qHL: <RSCAN1_base> + $060A_H$ + ($10_H$ × q),
RSCAN1RMDF0qHH: <RSCAN1_base> + $060B_H$ + ($10_H$ × q)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMDB3[7:0] | | | | | | | | RMDB2[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMDB1[7:0] | | | | | | | | RMDB0[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.238 RSCAN1RMDF0q Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RMDB3[7:0] | Receive Buffer Data Byte 3 |
| 23 to 16 | RMDB2[7:0] | Receive Buffer Data Byte 2 |
| 15 to 8 | RMDB1[7:0] | Receive Buffer Data Byte 1 |
| 7 to 0 | RMDB0[7:0] | Receive Buffer Data Byte 0<br>Data for a message stored in the receive buffer can be read. |

When the RMDLC[3:0] value in the RSCAN1RMPTRq register is smaller than $1000_B$, data bytes for which no data is set are read as $00_H$.

## 21.16.23 RSCAN1RMDF1q — Receive Buffer Data Field 1 Register (q = 0 to 15)

**Access:** RSCAN1RMDF1q register is a read-only register that can be read in 32-bit units.
RSCAN1RMDF1qL, RSCAN1RMDF1qH registers are read-only registers that can be read in 16-bit units.
RSCAN1RMDF1qLL, RSCAN1RMDF1qLH, RSCAN1RMDF1qHL, RSCAN1RMDF1qHH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1RMDF1q: <RSCAN1_base> + $060C_H$ + ($10_H$ × q)

RSCAN1RMDF1qL: <RSCAN1_base> + $060C_H$ + ($10_H$ × q),
RSCAN1RMDF1qH: <RSCAN1_base> + $060E_H$ + ($10_H$ × q)

RSCAN1RMDF1qLL: <RSCAN1_base> + $060C_H$ + ($10_H$ × q),
RSCAN1RMDF1qLH: <RSCAN1_base> + $060D_H$ + ($10_H$ × q),
RSCAN1RMDF1qHL: <RSCAN1_base> + $060E_H$ + ($10_H$ × q),
RSCAN1RMDF1qHH: <RSCAN1_base> + $060F_H$ + ($10_H$ × q)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMDB7[7:0] | | | | | | | | RMDB6[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMDB5[7:0] | | | | | | | | RMDB4[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.239 RSCAN1RMDF1q Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RMDB7[7:0] | Receive Buffer Data Byte 7 |
| 23 to 16 | RMDB6[7:0] | Receive Buffer Data Byte 6 |
| 15 to 8 | RMDB5[7:0] | Receive Buffer Data Byte 5 |
| 7 to 0 | RMDB4[7:0] | Receive Buffer Data Byte 4<br>Data for a message stored in the receive buffer can be read. |

When the RMDLC[3:0] value in the RSCAN1RMPTRq register is smaller than $1000_B$, data bytes for which no data is set are read as $00_H$.

## 21.16.24 RSCAN1RFCCx — Receive FIFO Buffer Configuration and Control Register (x = 0 to 7)

**Access:** RSCAN1RFCCx register can be read or written in 32-bit units.
RSCAN1RFCCxL register can be read or written in 16-bit units.
RSCAN1RFCCxLL, RSCAN1RFCCxLH registers can be read or written in 8-bit units.

**Address:** RSCAN1RFCCx: <RSCAN1_base> + $00B8_H$ + ($04_H$ × x)

RSCAN1RFCCxL: <RSCAN1_base> + $00B8_H$ + ($04_H$ × x)

RSCAN1RFCCxLL: <RSCAN1_base> + $00B8_H$ + ($04_H$ × x),
RSCAN1RFCCxLH: <RSCAN1_base> + $00B9_H$ + ($04_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFIGCV[2:0] | | | RFIM | — | RFDC[2:0] | | | — | — | — | — | — | — | RFIE | RFE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

**Table 21.240 RSCAN1RFCCx Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 15 to 13 | RFIGCV[2:0] | Receive FIFO Interrupt Request Timing Select<br>b15 b14 b13<br>0 0 0: When FIFO is 1/8 full.<br>0 0 1: When FIFO is 2/8 full.<br>0 1 0: When FIFO is 3/8 full.<br>0 1 1: When FIFO is 4/8 full.<br>1 0 0: When FIFO is 5/8 full.<br>1 0 1: When FIFO is 6/8 full.<br>1 1 0: When FIFO is 7/8 full.<br>1 1 1: When FIFO is full. |
| 12 | RFIM | Receive FIFO Interrupt Source Select<br>0: An interrupt occurs when the condition set by the RFIGCV[2:0] bits is met.<br>1: An interrupt occurs each time a message has been received. |
| 11 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 10 to 8 | RFDC[2:0] | Receive FIFO Buffer Depth Configuration<br>b10 b9 b8<br>0 0 0: 0 messages<br>0 0 1: 4 messages<br>0 1 0: 8 messages<br>0 1 1: 16 messages<br>1 0 0: 32 messages<br>1 0 1: 48 messages<br>1 1 0: 64 messages<br>1 1 1: 128 messages |
| 7 to 2 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 1 | RFIE | Receive FIFO Interrupt Enable<br>0: Receive FIFO interrupt is disabled.<br>1: Receive FIFO interrupt is enabled. |

Table 21.240 RSCAN1RFCCx Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | RFE | Receive FIFO Buffer Enable<br>0: No receive FIFO buffer is used.<br>1: Receive FIFO buffers are used. |

### RFIGCV[2:0] Bits

These bits are used to specify the number of received messages for generating a receive FIFO interrupt request when the RFIM bit is set to 0 with a fraction for the total number of buffers (the setting of RFDC[2:0]).

When the RFDC[2:0] bits are set to $001_B$ (4 messages), set the RFIGCV[2:0] bits to $001_B$, $011_B$, $101_B$, or $111_B$. Modify these bits only in global reset mode.

### RFIM Bit

This bit is used to select a FIFO interrupt source. Modify this bit only in global reset mode.

### RFDC[2:0] Bits

These bits are used to select the number of messages that can be stored in a single receive FIFO buffer. When these bits are set to $000_B$, no receive FIFO buffer should be used. Modify these bits only in global reset mode.

### RFIE Bit

Setting the RFIE bit to 1 enables receive FIFO interrupts. Modify this bit when the RFE bit set to 0 (no receive FIFO buffer is used).

### RFE Bit

Setting the RFE bit to 1 makes receive FIFO buffers available. Clearing this bit to 0 sets the RFEMP flag in the RSCAN1RFSTSx register to 1 (buffer empty). Modify this bit in global operating mode or global test mode.

## 21.16.25 RSCAN1RFSTSx — Receive FIFO Buffer Status Register (x = 0 to 7)

**Access:** RSCAN1RFSTSx register can be read or written in 32-bit units.
RSCAN1RFSTSxL register can be read or written in 16-bit units.
RSCAN1RFSTSxLL register can be read or written in 8-bit units.
RSCAN1RFSTSxLH register is a read-only register that can be read in 8-bit units.

**Address:** RSCAN1RFSTSx: <RSCAN1_base> + $00D8_H$ + ($04_H$ × x)

RSCAN1RFSTSxL: <RSCAN1_base> + $00D8_H$ + ($04_H$ × x)

RSCAN1RFSTSxLL: <RSCAN1_base> + $00D8_H$ + ($04_H$ × x),
RSCAN1RFSTSxLH: <RSCAN1_base> + $00D9_H$ + ($04_H$ × x)

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFMC[7:0] | | | | | | | | — | — | — | — | RFIF | RFMLT | RFFLL | RFEMP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W*1 | R/W*1 | R | R |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.241 RSCAN1RFSTSx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 15 to 8 | RFMC[7:0] | Receive FIFO Unread Message Counter<br>The number of unread messages stored in the receive FIFO buffer is displayed. |
| 7 to 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 | RFIF | Receive FIFO Interrupt Request Flag<br>0: No receive FIFO interrupt request is present.<br>1: A receive FIFO interrupt request is present. |
| 2 | RFMLT | Receive FIFO Message Lost Flag<br>0: No receive FIFO message is lost.<br>1: A receive FIFO message is lost. |
| 1 | RFFLL | Receive FIFO Buffer Full Status Flag<br>0: The receive FIFO buffer is not full.<br>1: The receive FIFO buffer is full. |
| 0 | RFEMP | Receive FIFO Buffer Empty Status Flag<br>0: The receive FIFO buffer contains unread message.<br>1: The receive FIFO buffer contains no unread message (buffer empty). |

### RFMC[7:0] Flags

This flag indicates the number of unread messages in the receive FIFO buffer. This flag becomes $00_H$ when the RFE bit in the RSCAN1RFCCx register is set to 0.

### RFIF Flag

This flag is set to 1 when the receive FIFO interrupt request generation conditions set by the RFIGCV[2:0] bits and the RFIM bit in the RSCAN1RFCCx register are met. This flag is cleared to 0 in global reset mode or by writing 0 to this flag. Modify this bit in global operating mode or global test mode.
To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared. When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

### RFMLT Flag

This flag is set to 1 when an attempt is made to store a new message while the receive FIFO buffer is full. In this case, the new message is discarded.

This flag is cleared to 0 in global reset mode or by writing 0 to this flag.

Modify this bit in global operating mode or global test mode.

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared. When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

### RFFLL Flag

This flag is set to 1 when the number of messages stored in the receive FIFO buffer matches the FIFO buffer depth set by the RFDC[2:0] bits in the RSCAN1RFCCx register.

If the number of messages stored in the receive FIFO buffer becomes smaller than the FIFO buffer depth set by the RFDC[2:0] bits, this flag is cleared to 0. This flag is also cleared to 0 when the RFE bit in the RSCAN1RFCCx register is set to 0 (no receive FIFO buffer is used) or in global reset mode.

### RFEMP Flag

This flag is set to 1 when all messages in the receive FIFO buffer have been read. This flag is also set to 1 when the RFE bit in the RSCAN1RFCCx register is 0 or in global reset mode.

This flag is cleared to 0 when even a single received message has been stored in the receive FIFO buffer.

**NOTE**

To clear the RFMLT or RFIF flag to 0, use a store instruction to write “0” to the given flag and “1” to the other flags.

## 21.16.26 RSCAN1RFPCTRx — Receive FIFO Buffer Pointer Control Register (x = 0 to 7)

**Access:** RSCAN1RFPCTRx register is a write-only register that can be written in 32-bit units.
RSCAN1RFPCTRxL register is a write-only register that can be written in 16-bit units.
RSCAN1RFPCTRxLL register is a write-only register that can be written in 8-bit units.

**Address:** RSCAN1RFPCTRx: <RSCAN1_base> + $00F8_H$ + ($04_H$ × x)

RSCAN1RFPCTRxL: <RSCAN1_base> + $00F8_H$ + ($04_H$ × x)

RSCAN1RFPCTRxLL: <RSCAN1_base> + $00F8_H$ + ($04_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RFPC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**Table 21.242 RSCAN1RFPCTRx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing, write the value after reset. |
| 7 to 0 | RFPC[7:0] | Receive FIFO Pointer Control<br>When these bits are set to $FF_H$, the read pointer moves to the next unread message in the receive FIFO buffer. |

### RFPC[7:0] Bits

When the RFPC[7:0] bits are set to $FF_H$, the read pointer moves to the next unread message in the receive FIFO buffer. At this time, the RFMC[7:0] (receive FIFO unread message counter) value in the RSCAN1RFSTSx register is decremented. Read the RSCAN1RFIDx, RSCAN1RFPTRx, RSCAN1RFDF0x, and RSCAN1RFDF1x registers to read messages in the receive FIFO buffer, and then write $FF_H$ to the RFPC[7:0] bits.

When writing $FF_H$ to these bits, make sure that the RFE bit in the RSCAN1RFCCx register is set to 1 (receive FIFO buffers are used) and the RFEMP flag in the RSCAN1RFSTSx register is 0 (the receive FIFO buffer contains unread messages).

## 21.16.27 RSCAN1RFIDx — Receive FIFO Buffer Access ID Register (x = 0 to 7)

**Access:** RSCAN1RFIDx register is a read-only register that can be read in 32-bit units.
RSCAN1RFIDxL, RSCAN1RFIDxH registers are read-only registers that can be read in 16-bit units.
RSCAN1RFIDxLL, RSCAN1RFIDxLH, RSCAN1RFIDxHL, RSCAN1RFIDxHH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1RFIDx: <RSCAN1_base> + $0E00_H$ + ($10_H$ × x)

RSCAN1RFIDxL: <RSCAN1_base> + $0E00_H$ + ($10_H$ × x),
RSCAN1RFIDxH: <RSCAN1_base> + $0E02_H$ + ($10_H$ × x)

RSCAN1RFIDxLL: <RSCAN1_base> + $0E00_H$ + ($10_H$ × x),
RSCAN1RFIDxLH: <RSCAN1_base> + $0E01_H$ + ($10_H$ × x),
RSCAN1RFIDxHL: <RSCAN1_base> + $0E02_H$ + ($10_H$ × x),
RSCAN1RFIDxHH: <RSCAN1_base> + $0E03_H$ + ($10_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFIDE | RFRTR | — | RFID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.243 RSCAN1RFIDx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | RFIDE | Receive FIFO Buffer IDE<br>0: Standard ID<br>1: Extended ID |
| 30 | RFRTR | Receive FIFO Buffer RTR<br>0: Data frame<br>1: Remote frame |
| 29 | Reserved | When read, the value after reset is returned. |
| 28 to 0 | RFID[28:0] | Receive FIFO Buffer ID Data<br>The standard ID or extended ID of received message can be read.<br>Read bits b10 to b0 for standard ID. Bits b28 to b11 are read as 0. |

### RFIDE Bit

This bit indicates the ID format (standard ID or extended ID) of the message stored in the receive FIFO buffer.

### RFRTR Bit

This bit indicates the frame format (data frame or remote frame) of the message stored in the receive FIFO buffer.

### RFID[28:0] Bits

These bits indicate the ID of the message stored in the receive FIFO buffer.

## 21.16.28 RSCAN1RFPTRx — Receive FIFO Buffer Access Pointer Register (x = 0 to 7)

**Access:** RSCAN1RFPTRx register is a read-only register that can be read in 32-bit units.
RSCAN1RFPTRxL, RSCAN1RFPTRxH registers are read-only registers that can be read in 16-bit units.
RSCAN1RFPTRxLL, RSCAN1RFPTRxLH, RSCAN1RFPTRxHL, RSCAN1RFPTRxHH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1RFPTRx: <RSCAN1_base> + $0E04_H$ + ($10_H$ × x)

RSCAN1RFPTRxL: <RSCAN1_base> + $0E04_H$ + ($10_H$ × x),
RSCAN1RFPTRxH: <RSCAN1_base> + $0E06_H$ + ($10_H$ × x)

RSCAN1RFPTRxLL: <RSCAN1_base> + $0E04_H$ + ($10_H$ × x),
RSCAN1RFPTRxLH: <RSCAN1_base> + $0E05_H$ + ($10_H$ × x),
RSCAN1RFPTRxHL: <RSCAN1_base> + $0E06_H$ + ($10_H$ × x),
RSCAN1RFPTRxHH: <RSCAN1_base> + $0E07_H$ + ($10_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFDLC[3:0] | | | | RFPTR[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFTS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.244 RSCAN1RFPTRx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | RFDLC[3:0] | Receive FIFO Buffer DLC Data<br>b31 b30 b29 b28<br>0 0 0 0: 0 data bytes<br>0 0 0 1: 1 data byte<br>0 0 1 0: 2 data bytes<br>0 0 1 1: 3 data bytes<br>0 1 0 0: 4 data bytes<br>0 1 0 1: 5 data bytes<br>0 1 1 0: 6 data bytes<br>0 1 1 1: 7 data bytes<br>1 X X X: 8 data bytes |
| 27 to 16 | RFPTR[11:0] | Receive FIFO Buffer Label Data<br>Label information of the received message can be read. |
| 15 to 0 | RFTS[15:0] | Receive FIFO Buffer Timestamp Data<br>Timestamp value of the received message can be read. |

### RFDLC[3:0] Bits

These bits contain the data length of the message stored in the receive FIFO buffer.

### RFPTR[11:0] Bits

These bits contain the label information of the message stored in the receive FIFO buffer.

### RFTS[15:0] Bits

These bits contain the timestamp value of the message stored in the receive FIFO buffer.

## 21.16.29 RSCAN1RFDF0x — Receive FIFO Buffer Access Data Field 0 Register (x = 0 to 7)

**Access:** RSCAN1RFDF0x register is a read-only register that can be read in 32-bit units.
RSCAN1RFDF0xL, RSCAN1RFDF0xH registers are read-only registers that can be read in 16-bit units.
RSCAN1RFDF0xLL, RSCAN1RFDF0xLH, RSCAN1RFDF0xHL, RSCAN1RFDF0xHH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1RFDF0x: <RSCAN1_base> + $0E08_H$ + ($10_H$ × x)

RSCAN1RFDF0xL: <RSCAN1_base> + $0E08_H$ + ($10_H$ × x),
RSCAN1RFDF0xH: <RSCAN1_base> + $0E0A_H$ + ($10_H$ × x)

RSCAN1RFDF0xLL: <RSCAN1_base> + $0E08_H$ + ($10_H$ × x),
RSCAN1RFDF0xLH: <RSCAN1_base> + $0E09_H$ + ($10_H$ × x),
RSCAN1RFDF0xHL: <RSCAN1_base> + $0E0A_H$ + ($10_H$ × x),
RSCAN1RFDF0xHH: <RSCAN1_base> + $0E0B_H$ + ($10_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFDB3[7:0] | | | | | | | | RFDB2[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFDB1[7:0] | | | | | | | | RFDB0[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.245 RSCAN1RFDF0x Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RFDB3[7:0] | Receive FIFO Buffer Data Byte 3 |
| 23 to 16 | RFDB2[7:0] | Receive FIFO Buffer Data Byte 2 |
| 15 to 8 | RFDB1[7:0] | Receive FIFO Buffer Data Byte 1 |
| 7 to 0 | RFDB0[7:0] | Receive FIFO Buffer Data Byte 0<br>Data for a message stored in the receive FIFO buffer can be read. |

When the RFDLC[3:0] value in the RSCAN1RFPTRx register is smaller than $1000_B$, data bytes for which no data is set are read as $00_H$.

## 21.16.30 RSCAN1RFDF1x — Receive FIFO Buffer Access Data Field 1 Register (x = 0 to 7)

**Access:** RSCAN1RFDF1x register is a read-only register that can be read in 32-bit units.
RSCAN1RFDF1xL, RSCAN1RFDF1xH registers are read-only registers that can be read in 16-bit units.
RSCAN1RFDF1xLL, RSCAN1RFDF1xLH, RSCAN1RFDF1xHL, RSCAN1RFDF1xHH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1RFDF1x: <RSCAN1_base> + $0E0C_H$ + ($10_H$ × x)

RSCAN1RFDF1xL: <RSCAN1_base> + $0E0C_H$ + ($10_H$ × x),
RSCAN1RFDF1xH: <RSCAN1_base> + $0E0E_H$ + ($10_H$ × x)

RSCAN1RFDF1xLL: <RSCAN1_base> + $0E0C_H$ + ($10_H$ × x),
RSCAN1RFDF1xLH: <RSCAN1_base> + $0E0D_H$ + ($10_H$ × x),
RSCAN1RFDF1xHL: <RSCAN1_base> + $0E0E_H$ + ($10_H$ × x),
RSCAN1RFDF1xHH: <RSCAN1_base> + $0E0F_H$ + ($10_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFDB7[7:0] | | | | | | | | RFDB6[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFDB5[7:0] | | | | | | | | RFDB4[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.246 RSCAN1RFDF1x Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RFDB7[7:0] | Receive FIFO Buffer Data Byte 7 |
| 23 to 16 | RFDB6[7:0] | Receive FIFO Buffer Data Byte 6 |
| 15 to 8 | RFDB5[7:0] | Receive FIFO Buffer Data Byte 5 |
| 7 to 0 | RFDB4[7:0] | Receive FIFO Buffer Data Byte 4<br>Data for a message stored in the receive FIFO buffer can be read. |

When the RFDLC[3:0] value in the RSCAN1RFPTRx register is smaller than $1000_B$, data bytes for which no data is set are read as $00_H$.

## 21.16.31 RSCAN1CFCCk — Transmit/receive FIFO Buffer Configuration and Control Register (k = 0 to 2)

**Access:** RSCAN1CFCCk register can be read or written in 32-bit units.
RSCAN1CFCCkL, RSCAN1CFCCkH registers can be read or written in 16-bit units.
RSCAN1CFCCkLL, RSCAN1CFCCkLH, RSCAN1CFCCkHL, RSCAN1CFCCkHH registers can be read or written in 8-bit units.

**Address:** RSCAN1CFCCk: <RSCAN1_base> + $0118_H$ + ($04_H$ × k)

RSCAN1CFCCkL: <RSCAN1_base> + $0118_H$ + ($04_H$ × k),
RSCAN1CFCCkH: <RSCAN1_base> + $011A_H$ + ($04_H$ × k)

RSCAN1CFCCkLL: <RSCAN1_base> + $0118_H$ + ($04_H$ × k),
RSCAN1CFCCkLH: <RSCAN1_base> + $0119_H$ + ($04_H$ × k),
RSCAN1CFCCkHL: <RSCAN1_base> + $011A_H$ + ($04_H$ × k),
RSCAN1CFCCkHH: <RSCAN1_base> + $011B_H$ + ($04_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFITT[7:0] | | | | | | | | CFTML[3:0] | | | | CFITR | CFITSS | CFM[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFIGCV[2:0] | | | CFIM | — | CFDC[2:0] | | | — | — | — | — | — | CFTXIE | CFRXIE | CFE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

**Table 21.247 RSCAN1CFCCk Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | CFITT[7:0] | Set a message transmission interval.<br>Set Value: $00_H$ to $FF_H$ |
| 23 to 20 | CFTML[3:0] | Transmit Buffer Link Configuration<br>Set the transmit buffer number to be linked to the transmit/receive FIFO buffer. |
| 19 | CFITR | Transmit/Receive FIFO Interval Timer Resolution<br>0: Clock dividing pclk/2 by (ITRCP [15:0] bits)<br>1: Clock dividing pclk/2 by (ITRCP [15:0] bits × 10) |
| 18 | CFITSS | Transmit/Receive FIFO Interval Timer Clock Source Select<br>0: Interval timer clock source selected by the CFITR bit<br>1: Interval timer clock source is the bit time clock for the channel to which the FIFO is linked. |
| 17, 16 | CFM[1:0] | Transmit/Receive FIFO Mode Select<br>b17 b16<br>0 0: Receive mode<br>0 1: Transmit mode<br>1 0: Setting prohibited<br>1 1: Setting prohibited |
| 15 to 13 | CFIGCV[2:0] | Transmit/Receive FIFO Receive Interrupt Request Timing Select<br>b15 b14 b13<br>0 0 0: When FIFO is 1/8 full.<br>0 0 1: When FIFO is 2/8 full.<br>0 1 0: When FIFO is 3/8 full.<br>0 1 1: When FIFO is 4/8 full.<br>1 0 0: When FIFO is 5/8 full.<br>1 0 1: When FIFO is 6/8 full.<br>1 1 0: When FIFO is 7/8 full.<br>1 1 1: When FIFO is full. |

Table 21.247 RSCAN1CFCCk Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 12 | CFIM | Transmit/Receive FIFO Interrupt Source Select<br>0:<br>• Receive mode<br>When the number of received messages has met the condition set by the CFIGCV[2:0] bits, a FIFO receive interrupt request is generated.<br>• Transmit mode<br>When the buffer becomes empty upon completion of message transmission, a FIFO transmit interrupt request is generated.<br>1:<br>• Receive mode<br>A FIFO receive interrupt request is generated each time a message has been received.<br>• Transmit mode<br>A FIFO transmit interrupt request is generated each time a message has been transmitted. |
| 11 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 10 to 8 | CFDC[2:0] | Transmit/Receive FIFO Buffer Depth Configuration<br>b10 b9 b8<br>0 0 0: 0 messages<br>0 0 1: 4 messages<br>0 1 0: 8 messages<br>0 1 1: 16 messages<br>1 0 0: 32 messages<br>1 0 1: 48 messages<br>1 1 0: 64 messages<br>1 1 1: 128 messages |
| 7 to 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 | CFTXIE | Transmit/Receive FIFO Transmit Interrupt Enable<br>0: Transmit/receive FIFO transmit interrupt is disabled.<br>1: Transmit/receive FIFO transmit interrupt is enabled. |
| 1 | CFRXIE | Transmit/Receive FIFO Receive Interrupt Enable<br>0: Transmit/receive FIFO receive interrupt is disabled.<br>1: Transmit/receive FIFO receive interrupt is enabled. |
| 0 | CFE | Transmit/Receive FIFO Buffer Enable<br>0: No transmit/receive FIFO buffer is used.<br>1: Transmit/receive FIFO buffers are used. |

### CFITT[7:0] Bits

These bits are used to set a message transmission interval when transmitting messages continuously from a transmit/receive FIFO buffer whose CFM[1:0] bits are set to $01_B$ (transmit mode).

Clear the CFE bit to 0 (no transmit/receive FIFO buffer is used) before modifying the CFITT[7:0] bits.

### CFTML[3:0] Bits

These bits are used to set the number of transmit buffer on the channel which will be linked to transmit/receive FIFO buffer k when the CFM[1:0] bits are set to $01_B$ (transmit mode). There are three transmit/receive FIFO buffers per channel, so channel number i of FIFO buffer k is calculated as i = k/3 (integer division). The actual assigned transmit buffer number p linked to FIFO buffer p will be ((16 × i) + CFTML[3:0]).

See **Table 21.214** and **Table 21.215**, as for the relationship between transmit/receive FIFO buffer k and transmit buffer p.

Setting the CFDC[2:0] bits to $001_B$ or more enables the setting of the CFTML[3:0] bits.

Do not link to any transmit buffer which is already allocated to a transmit queue on the identical channel or to another transmit/receive FIFO buffer. Modify these bits only in global reset mode.

#### CFITR Bit

This bit is enabled when the CFITSS bit is 0.

When this bit is 0, the interval timer clock source is the pclk/2 clock divided by the value of the ITRCP[15:0] bits in the RSCAN1GCFG register.

When this bit is 1, the interval timer clock source is the pclk/2 clock divided by the value of the ITRCP[15:0] bits in the RSCAN1GCFG register × 10.

Modify this bit while the CFE bit is set to 0 (no transmit/receive FIFO buffer is used).

#### CFITSS Bit

When this bit is 0, the clock selected by the CFITR bit is the count source of the interval timer.

When this bit is 1, the bit time clock of the channel to which the FIFO is linked is the count source of the interval timer.

Modify this bit while the CFE bit is set to 0 (no transmit/receive FIFO buffer is used).

#### CFM[1:0] Bits

These bits are used to select transmit/receive FIFO mode. Modify these bits only in global reset mode.

#### CFIGCV[2:0] Bits

These bits are used to specify the number of received messages for generating a transmit/receive FIFO receive interrupt request when the CFM[1:0] bits are set to $00_B$ (receive mode) and the CFIM bit is set to 0 with a fraction for the total number of buffers (the setting of CFDC[2:0]).

When the CFDC[2:0] bits are set to $001_B$ (4 messages), set the CFIGCV[2:0] bits to $001_B$, $011_B$, $101_B$, or $111_B$.

Modify these bits only in global reset mode.

#### CFIM Bit

This bit is used to select a transmit/receive FIFO interrupt source. Modify this bit only in global reset mode.

#### CFDC[2:0] Bits

These bits are used to set the number of messages that can be stored in a single transmit/receive FIFO buffer. When these bits are set to $000_B$, do not use a transmit/receive FIFO buffer. Modify these bits only in global reset mode.

#### CFTXIE Bit

When this bit is set to 1 and the CFTXIF flag in the RSCAN1CFSTSk register is set to 1, a transmit/receive FIFO transmit interrupt request is generated.

Modify this bit with the CFE bit set to 0 (no transmit/receive FIFO buffer is used).

### CFRXIE Bit

When this bit is set to 1 and the CFRXIF flag in the RSCAN1CFSTSk register is set to 1, a transmit/receive FIFO receive interrupt request is generated.

Modify this bit with the CFE bit set to 0.

### CFE Bit

Setting this bit to 1 makes transmit/receive FIFO buffers available.

When this bit is set to 0 in transmit mode, if a message in the transmit/receive FIFO buffer is being transmitted or will be transmitted next, the transmit/receive FIFO buffer becomes empty after completion of transmission of that message, or upon detection of a CAN bus error, or arbitration-lost. In other cases or in receive mode, the transmit/receive FIFO buffer becomes empty immediately.

This bit is cleared to 0 when the following conditions are met.

- Receive mode: Global reset mode
- Transmit mode: Channel reset mode

Modify this bit in the following mode.

- Receive mode: Global operating mode or global test mode
- Transmit mode: Channel communication mode or channel halt mode

## 21.16.32 RSCAN1CFSTSk — Transmit/receive FIFO Buffer Status Register (k = 0 to 2)

**Access:** RSCAN1CFSTSk register can be read or written in 32-bit units.
RSCAN1CFSTSkL register can be read or written in 16-bit units.
RSCAN1CFSTSkLL register can be read or written in 8-bit units.
RSCAN1CFSTSkLH register is a read-only register that can be read in 8-bit units.

**Address:** RSCAN1CFSTSk: <RSCAN1_base> + $0178_H$ + ($04_H$ × k)

RSCAN1CFSTSkL: <RSCAN1_base> + $0178_H$ + ($04_H$ × k)

RSCAN1CFSTSkLL: <RSCAN1_base> + $0178_H$ + ($04_H$ × k),
RSCAN1CFSTSkLH: <RSCAN1_base> + $0179_H$ + ($04_H$ × k)

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFMC[7:0] | | | | | | | | — | — | — | CFTXIF | CFRXIF | CFMLT | CFFLL | CFEMP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W*1 | R/W*1 | R/W*1 | R | R |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.248 RSCAN1CFSTSk Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 15 to 8 | CFMC[7:0] | Transmit/Receive FIFO Message Counter<br>The number of messages stored in the transmit/receive FIFO buffer. |
| 7 to 5 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 4 | CFTXIF | Transmit/Receive FIFO Transmit Interrupt Request Flag<br>0: No transmit/receive FIFO transmit interrupt request is present.<br>1: A transmit/receive FIFO transmit interrupt request is present. |
| 3 | CFRXIF | Transmit/Receive FIFO Receive Interrupt Request Flag<br>0: No transmit/receive FIFO receive interrupt request is present.<br>1: A transmit/receive FIFO receive interrupt request is present. |
| 2 | CFMLT | Transmit/Receive FIFO Message Lost Flag<br>0: No transmit/receive FIFO message is lost.<br>1: A transmit/receive FIFO message is lost. |
| 1 | CFFLL | Transmit/Receive FIFO Buffer Full Status Flag<br>0: The transmit/receive FIFO buffer is not full.<br>1: The transmit/receive FIFO buffer is full. |
| 0 | CFEMP | Transmit/Receive FIFO Buffer Empty Status Flag<br>0: The transmit/receive FIFO buffer contains messages.<br>1: The transmit/receive FIFO buffer contains no message (buffer empty). |

### CFMC[7:0] Bits

The CFMC[7:0] bits indicate the following values that depend on the setting of the CFM[1:0] bits in the RSCAN1CFCCk register.

- When CFM[1:0] value is $01_B$ (transmit mode): Number of untransmitted messages in the buffer

- When CFM[1:0] value is $00_B$ (receive mode): Number of unread received messages in the buffer

These bits are cleared to 0 when any of the following conditions is met.

- When CFM[1:0] value is $00_B$: In global reset mode
- When CFM[1:0] value is $01_B$: In channel reset mode

#### CFTXIF Flag

The CFTXIF flag is set to 1 when any of the following conditions is met.

- When the CFM[1:0] bits are set to $01_B$ or $10_B$, and the factor selected by the CFIM bit in the RSCAN1CFCCk register occurs

The CFTXIF flag is cleared to 0 when any of the following conditions is met.

- When 0 is written to the CFTXIF flag
- When the CFM[1:0] bits are set to $00_B$: In global reset mode
- When the CFM[1:0] bits are set to $01_B$ or $10_B$: In channel reset mode

Write 0 to this flag in global operating mode or global test mode.

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared. When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

#### CFRXIF Flag

The CFRXIF flag is set to 1 when any of the following conditions is met.

- When the CFM[1:0] bits are set to $00_B$ or $10_B$, and the factor selected by the CFIM bit in the RSCAN1CFCCk register occurs

The CFRXIF flag is cleared to 0 when any of the following conditions is met.

- When 0 is written to the CFRXIF flag
- When the CFM[1:0] bits are set to $00_B$: In global reset mode
- When the CFM[1:0] bits are set to $01_B$ or $10_B$: In channel reset mode

Write 0 to this flag in global operating mode or global test mode.

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared.

When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

#### CFMLT Flag

The CFMLT flag is set to 1 when any of the following conditions is met.

- When an attempt is made to store a new message while the transmit/receive FIFO buffer is full. In this case, the new message is discarded.

The CFMLT flag is cleared to 0 when any of the following conditions is met.

- When 0 is written to the CFMLT flag
- When the CFM[1:0] bits are set to $00_B$: In global reset mode
- When the CFM[1:0] bits are set to $01_B$ or $10_B$: In channel reset mode

Write 0 to this flag in global operating mode or global test mode

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared. When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

### CFFLL Flag

The CFFLL flag is set to 1 when any of the following conditions is met.

- When the number of messages stored in the transmit/receive FIFO buffer matches the FIFO buffer depth set by the CFDC[2:0] bits in the RSCAN1CFCCk register.

The CFFLL flag is cleared to 0 when any of the following conditions is met.

- When the number of messages stored in the transmit/receive FIFO buffer becomes smaller than the FIFO buffer depth set by the CFDC[2:0] bits.
- When the CFE bit in the RSCAN1CFCCk register is 0 (no transmit/receive FIFO buffer is used): When not performing transmit abort
- When the CFM[1:0] bits are set to $00_B$: In global reset mode
- When the CFM[1:0] bits are set to $01_B$ or $10_B$: In channel reset mode

### CFEMP Flag

The CFEMP flag is set to 1 when any of the following conditions is met.

- When the CFM[1:0] bits are set to $00_B$: All messages have been read, or in global reset mode
- When the CFM[1:0] bits are set to $01_B$ or $10_B$: All messages have been transmitted, or in channel reset mode
- When the CFE bit is 0 (no transmit/receive FIFO buffer is used): When not performing transmit abort

The CFEMP flag is cleared to 0 when any of the following conditions is met.

- When the CFM[1:0] bits are set to $00_B$ or $10_B$: At least one received message has been stored in the transmit/receive FIFO buffer.
- When the CFM[1:0] bits are set to $01_B$: A value of $FF_H$ has been written to the RSCAN1CFPCTRk register after data was written to the RSCAN1CFIDk, RSCAN1CFPTRk, RSCAN1CFDF0k, and RSCAN1CFDF1k registers.

**NOTE**

To clear CFTXIF, CFRXIF, or CFMLT flag to 0, the program must write 0. When writing, use a store instruction to write “0” to the given flag and “1” to other flags.

## 21.16.33 RSCAN1CFPCTRk — Transmit/receive FIFO Buffer Pointer Control Register (k = 0 to 2)

**Access:** RSCAN1CFPCTRk register is a write-only register that can be written in 32-bit units.
RSCAN1CFPCTRkL register is a write-only register that can be written in 16-bit units.
RSCAN1CFPCTRkLL register is a write-only register that can be written in 8-bit units.

**Address:** RSCAN1CFPCTRk: <RSCAN1_base> + $01D8_H$ + ($04_H$ × k)

RSCAN1CFPCTRkL: <RSCAN1_base> + $01D8_H$ + ($04_H$ × k)

RSCAN1CFPCTRkLL: <RSCAN1_base> + $01D8_H$ + ($04_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CFPC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**Table 21.249 RSCAN1CFPCTRk Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing, write the value after reset. |
| 7 to 0 | CFPC[7:0] | Transmit/Receive FIFO Pointer Control<br>• Receive mode:<br>Writing $FF_H$ to these bits moves the read pointer to the next unread message in the transmit/receive FIFO buffer.<br>• Transmit mode:<br>Writing $FF_H$ to these bits moves the write pointer to the next stage of the transmit/receive FIFO buffer. |

### CFPC[7:0] Bits

- Receive mode (CFM[1:0] value in the RSCAN1CFCCk register is $00_B$):
  Writing $FF_H$ to the CFPC[7:0] bits moves the read pointer to the next unread message in the transmit/receive FIFO buffer. At this time, the CFMC[7:0] value (transmit/receive FIFO message counter) in the RSCAN1CFSTSk register is decremented. Read the RSCAN1CFIDk, RSCAN1CFPTRk, RSCAN1CFDF0k, and RSCAN1CFDF1k registers to read messages from the transmit/receive FIFO buffer, and then write $FF_H$ to the CFPC[7:0] bits.
  When writing $FF_H$ to these bits, make sure that the CFE bit in the RSCAN1CFCCk register is set to 1 (transmit/receive FIFO buffers are used) and the CFEMP flag in the RSCAN1CFSTSk register is 0 (the transmit/receive FIFO buffer contains messages).

- Transmit mode (CFM[1:0] value in the RSCAN1CFCCk register is $01_B$):
  Writing $FF_H$ to the CFPC[7:0] bits stores the data written to the RSCAN1CFIDk, RSCAN1CFPTRk, RSCAN1CFDF0k, and RSCAN1CFDF1k registers in the transmit/receive FIFO buffer and moves the write pointer to the next stage of the transmit/receive FIFO buffer. At this time, the CFMC[7:0] value is incremented. Write transmit messages to the RSCAN1CFIDk, RSCAN1CFPTRk, RSCAN1CFDF0k, and RSCAN1CFDF1k registers before writing $FF_H$ to the CFPC[7:0] bits.
  When writing $FF_H$ to these bits, make sure that the CFE bit in the RSCAN1CFCCk register is set to 1 and the CFFLL flag in the RSCAN1CFSTSk register is 0 (the transmit/receive FIFO buffer is not full).

## 21.16.34 RSCAN1CFIDk — Transmit/receive FIFO Buffer Access ID Register (k = 0 to 2)

**Access:** RSCAN1CFIDk register can be read or written in 32-bit units.
RSCAN1CFIDkL, RSCAN1CFIDkH registers can be read or written in 16-bit units.
RSCAN1CFIDkLL, RSCAN1CFIDkLH, RSCAN1CFIDkHL, RSCAN1CFIDkHH registers can be read or written in 8-bit units.

**Address:** RSCAN1CFIDk: <RSCAN1_base> + $0E80_H$ + ($10_H$ × k)

RSCAN1CFIDkL: <RSCAN1_base> + $0E80_H$ + ($10_H$ × k),
RSCAN1CFIDkH: <RSCAN1_base> + $0E82_H$ + ($10_H$ × k)

RSCAN1CFIDkLL: <RSCAN1_base> + $0E80_H$ + ($10_H$ × k),
RSCAN1CFIDkLH: <RSCAN1_base> + $0E81_H$ + ($10_H$ × k),
RSCAN1CFIDkHL: <RSCAN1_base> + $0E82_H$ + ($10_H$ × k),
RSCAN1CFIDkHH: <RSCAN1_base> + $0E83_H$ + ($10_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFIDE | CFRTR | THLEN | CFID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.250 RSCAN1CFIDk Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | CFIDE | Transmit/Receive FIFO Buffer IDE<br>0: Standard ID<br>1: Extended ID |
| 30 | CFRTR | Transmit/Receive FIFO Buffer RTR<br>0: Data frame<br>1: Remote frame |
| 29 | THLEN | Transmit History Data Store Enable<br>This bit is valid only when the CFM[1:0] value is $01_B$ (transmit mode).<br>0: Transmit history data is not stored in the buffer.<br>1: Transmit history data is stored in the buffer. |
| 28 to 0 | CFID[28:0] | Transmit/Receive FIFO Buffer ID Data<br>• When CFM[1:0] value is $01_B$ (transmit mode):<br>Set standard ID or extended ID. For standard ID, write an ID to bits 10 to 0 and write 0 to bits 28 to 11.<br>• When CFM[1:0] value is $00_B$ (receive mode):<br>Standard ID or extended ID in the received message can be read. For standard ID, read bits 10 to 0. Bits 28 to 11 are read as 0. |

This register is writable only when the CFM[1:0] value in the RSCAN1CFCCk register is $01_B$ (transmit mode). This register is readable only when the CFM[1:0] value is $00_B$ (receive mode).

#### CFIDE Bit

This bit indicates the ID format (standard ID or extended ID) of the received message stored in the transmit/receive FIFO buffer when the CFM[1:0] value is $00_B$. When the CFM[1:0] value is $01_B$, this bit is used to set the ID format of the message to be transmitted from the transmit/receive FIFO buffer.

#### CFRTR Bit

This bit indicates the data format (data frame or remote frame) of the received message stored in the transmit/receive FIFO buffer when the CFM[1:0] value is $00_B$. When the CFM[1:0] value is $01_B$, this bit is used to set the data format of the message to be transmitted from the transmit/receive FIFO buffer.

#### THLEN Bit

When this bit is set to 1, the transmit history data (label information, buffer number, and buffer type) of transmit messages is stored in the transmit history buffer after transmission is completed.

This bit is enabled when the CFM[1:0] value is $01_B$ (transmit mode).

#### CFID[28:0] Bits

These bits contain the ID of the received message stored in the transmit/receive FIFO buffer when the CFM[1:0] value is $00_B$.

When the CFM[1:0] value is $01_B$, these bits are used to set the ID of the message to be transmitted from the transmit/receive FIFO buffer.

## 21.16.35 RSCAN1CFPTRk — Transmit/receive FIFO Buffer Access Pointer Register (k = 0 to 2)

**Access:** RSCAN1CFPTRk register can be read or written in 32-bit units.
RSCAN1CFPTRkL, RSCAN1CFPTRkH registers can be read or written in 16-bit units.
RSCAN1CFPTRkLL, RSCAN1CFPTRkLH, RSCAN1CFPTRkHL, RSCAN1CFPTRkHH registers can be read or written in 8-bit units.

**Address:** RSCAN1CFPTRk: <RSCAN1_base> + $0E84_H$ + ($10_H$ × k)

RSCAN1CFPTRkL: <RSCAN1_base> + $0E84_H$ + ($10_H$ × k),
RSCAN1CFPTRkH: <RSCAN1_base> + $0E86_H$ + ($10_H$ × k)

RSCAN1CFPTRkLL: <RSCAN1_base> + $0E84_H$ + ($10_H$ × k),
RSCAN1CFPTRkLH: <RSCAN1_base> + $0E85_H$ + ($10_H$ × k),
RSCAN1CFPTRkHL: <RSCAN1_base> + $0E86_H$ + ($10_H$ × k),
RSCAN1CFPTRkHH: <RSCAN1_base> + $0E87_H$ + ($10_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFDLC[3:0] | | | | CFPTR[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFTS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.251 RSCAN1CFPTRk Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | CFDLC[3:0] | Transmit/Receive FIFO Buffer DLC Data<br>b31 b30 b29 b28<br>0 0 0 0: 0 data bytes<br>0 0 0 1: 1 data byte<br>0 0 1 0: 2 data bytes<br>0 0 1 1: 3 data bytes<br>0 1 0 0: 4 data bytes<br>0 1 0 1: 5 data bytes<br>0 1 1 0: 6 data bytes<br>0 1 1 1: 7 data bytes<br>1 X X X: 8 data bytes |
| 27 to 16 | CFPTR[11:0] | Transmit/Receive FIFO Buffer Label Data<br>• When CFM[1:0] value is $01_B$ (transmit mode):<br>Set the label information to be stored in the transmit history buffer.<br>Only bits CFPTR[7:0] are valid.<br>• When CFM[1:0] value is $00_B$ (receive mode):<br>The label information of the received message can be read. |
| 15 to 0 | CFTS[15:0] | Transmit/Receive FIFO Buffer Timestamp Data<br>These bits are valid only when the CFM[1:0] value is $00_B$ (receive mode).<br>The timestamp value of the received message can be read. |

This register is writable only when the CFM[1:0] value in the RSCAN1CFCCk register is $01_B$ (transmit mode). This register is readable only when the CFM[1:0] value is $00_B$ (receive mode).

#### CFDLC[3:0] Bits

These bits indicate the data length of the received message stored in the transmit/receive FIFO buffer when the CFM[1:0] value is $00_B$. When the CFM[1:0] value is $01_B$, these bits are used to set the data length of the message to be transmitted from the transmit/receive FIFO buffer. If the data length is set to 9 bytes or more, the actual transmit data defaults to 8 bytes.

#### CFPTR[11:0] Bits

These bits indicate the label information attached to the received message stored in the transmit/receive FIFO buffer when the CFM[1:0] value is $00_B$. When the CFM[1:0] value is $01_B$, the CFPTR[7:0] value is stored in the transmit history buffer when message transmission has been completed.

#### CFTS[15:0] Bits

These bits indicate the timestamp value of the message stored in the transmit/receive FIFO buffer.

These bits are valid when the CFM[1:0] value is $00_B$.

## 21.16.36 RSCAN1CFDF0k — Transmit/receive FIFO Buffer Access Data Field 0 Register (k = 0 to 2)

**Access:** RSCAN1CFDF0k register can be read or written in 32-bit units.
RSCAN1CFDF0kL, RSCAN1CFDF0kH registers can be read or written in 16-bit units.
RSCAN1CFDF0kLL, RSCAN1CFDF0kLH, RSCAN1CFDF0kHL, RSCAN1CFDF0kHH registers can be read or written in 8-bit units.

**Address:** RSCAN1CFDF0k: <RSCAN1_base> + $0E88_H$ + ($10_H$ × k)

RSCAN1CFDF0kL: <RSCAN1_base> + $0E88_H$ + ($10_H$ × k),
RSCAN1CFDF0kH: <RSCAN1_base> + $0E8A_H$ + ($10_H$ × k)

RSCAN1CFDF0kLL: <RSCAN1_base> + $0E88_H$ + ($10_H$ × k),
RSCAN1CFDF0kLH: <RSCAN1_base> + $0E89_H$ + ($10_H$ × k),
RSCAN1CFDF0kHL: <RSCAN1_base> + $0E8A_H$ + ($10_H$ × k),
RSCAN1CFDF0kHH: <RSCAN1_base> + $0E8B_H$ + ($10_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFDB3[7:0] | | | | | | | | CFDB2[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFDB1[7:0] | | | | | | | | CFDB0[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.252 RSCAN1CFDF0k Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | CFDB3[7:0] | Transmit/Receive FIFO Buffer Data Byte 3<br>Transmit/Receive FIFO Buffer Data Byte 2<br>Transmit/Receive FIFO Buffer Data Byte 1<br>Transmit/Receive FIFO Buffer Data Byte 0<br>• When CFM[1:0] value is $01_B$ (transmit mode): Set the transmit/receive FIFO buffer data.<br>• When CFM[1:0] value is $00_B$ (receive mode): The message data stored in the transmit/receive FIFO buffer can be read. |
| 23 to 16 | CFDB2[7:0] | |
| 15 to 8 | CFDB1[7:0] | |
| 7 to 0 | CFDB0[7:0] | |

This register is writable only when the CFM[1:0] value in the RSCAN1CFCCk register is $01_B$ (transmit mode).

This register is readable only when the CFM[1:0] value is $00_B$ (receive mode). When the CFDLC[3:0] value in the RSCAN1CFPTRk register is smaller than $1000_B$, data bytes for which no data is set are read as $00_H$.

## 21.16.37 RSCAN1CFDF1k — Transmit/receive FIFO Buffer Access Data Field 1 Register (k = 0 to 2)

**Access:** RSCAN1CFDF1k register can be read or written in 32-bit units.
RSCAN1CFDF1kL, RSCAN1CFDF1kH registers can be read or written in 16-bit units.
RSCAN1CFDF1kLL, RSCAN1CFDF1kLH, RSCAN1CFDF1kHL, RSCAN1CFDF1kHH registers can be read or written in 8-bit units.

**Address:** RSCAN1CFDF1k: <RSCAN1_base> + $0E8C_H$ + ($10_H$ × k)

RSCAN1CFDF1kL: <RSCAN1_base> + $0E8C_H$ + ($10_H$ × k),
RSCAN1CFDF1kH: <RSCAN1_base> + $0E8E_H$ + ($10_H$ × k)

RSCAN1CFDF1kLL: <RSCAN1_base> + $0E8C_H$ + ($10_H$ × k),
RSCAN1CFDF1kLH: <RSCAN1_base> + $0E8D_H$ + ($10_H$ × k),
RSCAN1CFDF1kHL: <RSCAN1_base> + $0E8E_H$ + ($10_H$ × k),
RSCAN1CFDF1kHH: <RSCAN1_base> + $0E8F_H$ + ($10_H$ × k)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFDB7[7:0] | | | | | | | | CFDB6[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFDB5[7:0] | | | | | | | | CFDB4[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.253 RSCAN1CFDF1k Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | CFDB7[7:0] | Transmit/Receive FIFO Buffer Data Byte 7 |
| 23 to 16 | CFDB6[7:0] | Transmit/Receive FIFO Buffer Data Byte 6 |
| 15 to 8 | CFDB5[7:0] | Transmit/Receive FIFO Buffer Data Byte 5 |
| 7 to 0 | CFDB4[7:0] | Transmit/Receive FIFO Buffer Data Byte 4<br>• When CFM[1:0] value is $01_B$ (transmit mode): Set the transmit/receive FIFO buffer data.<br>• When CFM[1:0] value is $00_B$ (receive mode): The message data stored in the transmit/receive FIFO buffer can be read. |

This register is writable only when the CFM[1:0] value in the RSCAN1CFCCk register is $01_B$ (transmit mode).

This register is readable only when the CFM[1:0] value is $00_B$ (receive mode). When the CFDLC[3:0] value in the RSCAN1CFPTRk register is smaller than $1000_B$, data bytes for which no data is set are read as $00_H$.

## 21.16.38 RSCAN1FESTS — FIFO Empty Status Register

**Access:** RSCAN1FESTS register is a read-only register that can be read in 32-bit units.
RSCAN1FESTSL register is a read-only register that can be read in 16-bit units.
RSCAN1FESTSLL, RSCAN1FESTSLH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1FESTS: <RSCAN1_base> + $0238_H$

RSCAN1FESTSL: <RSCAN1_base> + $0238_H$

RSCAN1FESTSLL: <RSCAN1_base> + $0238_H$,
RSCAN1FESTSLH: <RSCAN1_base> + $0239_H$

**Value after reset:** $0001\ FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | CF2EMP | CF1EMP | CF0EMP | RF7EMP | RF6EMP | RF5EMP | RF4EMP | RF3EMP | RF2EMP | RF1EMP | RF0EMP |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.254 RSCAN1FESTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When read, the value after reset is returned. |
| 10 | CF2EMP | Transmit/Receive FIFO Buffer Empty Status Flag<br>0: Transmit/receive FIFO buffer k contains a message.<br>1: Transmit/receive FIFO buffer k contains no message.<br>(k = 0 to 2) |
| 9 | CF1EMP | |
| 8 | CF0EMP | |
| 7 | RF7EMP | Receive FIFO Buffer Empty Status Flag<br>0: Receive FIFO buffer x contains an unread message.<br>1: Receive FIFO buffer x contains no unread message (buffer empty).<br>(x = 0 to 7) |
| 6 | RF6EMP | |
| 5 | RF5EMP | |
| 4 | RF4EMP | |
| 3 | RF3EMP | |
| 2 | RF2EMP | |
| 1 | RF1EMP | |
| 0 | RF0EMP | |

The RSCAN1FESTS register is set to $0001\ FFFF_H$ in global reset mode.

### CFkEMP Flag (k = 0 to 2)

The CFkEMP flag is set to 1 when the CFEMP flag in the RSCAN1CFSTSk register is set to 1 (the transmit/receive FIFO buffer contains no message (buffer empty)). When the CFEMP flag is cleared to 0 (the transmit/receive FIFO buffer contains messages), the CFkEMP flag is cleared to 0.

### RFxEMP Flag (x = 0 to 7)

The RFxEMP flag is set to 1 when the RFEMP flag in the RSCAN1RFSTSx register is set to 1 (the receive FIFO buffer contains no unread message). When the RFEMP flag is cleared to 0 (the receive FIFO buffer contains unread messages), the RFxEMP flag is cleared to 0.

## 21.16.39 RSCAN1FFSTS — FIFO Full Status Register

**Access:** RSCAN1FFSTS register is a read-only register that can be read in 32-bit units.
RSCAN1FFSTSL register is a read-only register that can be read in 16-bit units.
RSCAN1FFSTSLL, RSCAN1FFSTSLH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1FFSTS: <RSCAN1_base> + $023C_H$

RSCAN1FFSTSL: <RSCAN1_base> + $023C_H$

RSCAN1FFSTSLL: <RSCAN1_base> + $023C_H$,
RSCAN1FFSTSLH: <RSCAN1_base> + $023D_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | CF2FLL | CF1FLL | CF0FLL | RF7FLL | RF6FLL | RF5FLL | RF4FLL | RF3FLL | RF2FLL | RF1FLL | RF0FLL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.255 RSCAN1FFSTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When read, the value after reset is returned. |
| 10 | CF2FLL | Transmit/Receive FIFO Buffer Full Status Flag<br>0: Transmit/receive buffer k is not full.<br>1: Transmit/receive buffer k is full.<br>(k = 0 to 2) |
| 9 | CF1FLL | |
| 8 | CF0FLL | |
| 7 | RF7FLL | Receive FIFO Buffer Full Status Flag<br>0: Receive FIFO buffer x is not full.<br>1: Receive FIFO buffer x is full.<br>(x = 0 to 7) |
| 6 | RF6FLL | |
| 5 | RF5FLL | |
| 4 | RF4FLL | |
| 3 | RF3FLL | |
| 2 | RF2FLL | |
| 1 | RF1FLL | |
| 0 | RF0FLL | |

The RSCAN1FFSTS register is cleared to $0000\ 0000_H$ in global reset mode.

### CFkFLL Flag (k = 0 to 2)

The CFkFLL flag is set to 1 when the CFFLL flag in the RSCAN1CFSTSk register is set to 1 (the transmit/receive FIFO buffer is full). When the CFFLL flag is cleared to 0 (the transmit/receive FIFO buffer is not full), the CFkFLL flag is cleared to 0.

### RFxFLL Flag (x = 0 to 7)

The RFxFLL flag is set to 1 when the RFFLL flag in the RSCAN1RFSTSx register is set to 1 (the receive FIFO buffer is full). When the RFFLL flag is cleared to 0 (the receive FIFO buffer is not full), the RFxFLL flag is cleared to 0.

## 21.16.40 RSCAN1FMSTS — FIFO Message Lost Status Register

**Access:** RSCAN1FMSTS register is a read-only register that can be read in 32-bit units.
RSCAN1FMSTSL register is a read-only register that can be read in 16-bit units.
RSCAN1FMSTSLL, RSCAN1FMSTSLH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1FMSTS: <RSCAN1_base> + $0240_H$

RSCAN1FMSTSL: <RSCAN1_base> + $0240_H$

RSCAN1FMSTSLL: <RSCAN1_base> + $0240_H$,
RSCAN1FMSTSLH: <RSCAN1_base> + $0241_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | CF2MLT | CF1MLT | CF0MLT | RF7MLT | RF6MLT | RF5MLT | RF4MLT | RF3MLT | RF2MLT | RF1MLT | RF0MLT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.256 RSCAN1FMSTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When read, the value after reset is returned. |
| 10 | CF2MLT | Transmit/Receive FIFO Buffer Message Lost Status Flag<br>0: No transmit/receive FIFO buffer k message is lost.<br>1: A transmit/receive FIFO buffer k message is lost.<br>(k = 0 to 2) |
| 9 | CF1MLT | |
| 8 | CF0MLT | |
| 7 | RF7MLT | Receive FIFO Buffer Message Lost Status Flag<br>0: No receive FIFO buffer x message is lost.<br>1: A receive FIFO buffer x message is lost.<br>(x = 0 to 7) |
| 6 | RF6MLT | |
| 5 | RF5MLT | |
| 4 | RF4MLT | |
| 3 | RF3MLT | |
| 2 | RF2MLT | |
| 1 | RF1MLT | |
| 0 | RF0MLT | |

The RSCAN1FMSTS register is cleared to $0000\ 0000_H$ in global reset mode.

### CFkMLT Flag (k = 0 to 2)

The CFkMLT flag is set to 1 when the CFMLT flag in the RSCAN1CFSTSk register is set to 1 (a transmit/receive FIFO message is lost). When the CFMLT flag is cleared to 0, the CFkMLT flag is cleared to 0.

### RFxMLT Flag (x = 0 to 7)

The RFxMLT flag is set to 1 when the RFMLT flag in the RSCAN1RFSTSx register is set to 1 (a receive FIFO message is lost). When the RFMLT flag is cleared to 0, the RFxMLT flag is cleared to 0.

## 21.16.41 RSCAN1RFISTS — Receive FIFO Buffer Interrupt Flag Status Register

**Access:** RSCAN1RFISTS register is a read-only register that can be read in 32-bit units.
RSCAN1RFISTSL register is a read-only register that can be read in 16-bit units.
RSCAN1RFISTSLL register is a read-only register that can be read in 8-bit units.

**Address:** RSCAN1RFISTS: <RSCAN1_base> + $0244_H$

RSCAN1RFISTSL: <RSCAN1_base> + $0244_H$

RSCAN1RFISTSLL: <RSCAN1_base> + $0244_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RF7IF | RF6IF | RF5IF | RF4IF | RF3IF | RF2IF | RF1IF | RF0IF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.257 RSCAN1RFISTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned. |
| 7 | RF7IF | Receive FIFO Buffer Interrupt Request Status Flag<br>0: No receive FIFO buffer x interrupt request is present.<br>1: A receive FIFO buffer x interrupt request is present.<br>(x = 0 to 7) |
| 6 | RF6IF | |
| 5 | RF5IF | |
| 4 | RF4IF | |
| 3 | RF3IF | |
| 2 | RF2IF | |
| 1 | RF1IF | |
| 0 | RF0IF | |

The RSCAN1RFISTS register is cleared to $0000\ 0000_H$ in global reset mode.

### RFxIF Flag (x = 0 to 7)

The RFxIF flag is set to 1 when the RFIF flag in the RSCAN1RFSTSx register is set to 1 (a receive FIFO interrupt request is present). When the RFIF flag is cleared to 0, the RFxIF flag is cleared to 0.

## 21.16.42 RSCAN1CFRISTS — Transmit/receive FIFO Buffer Receive Interrupt Flag Status Register

**Access:** RSCAN1CFRISTS register is a read-only register that can be read in 32-bit units.
RSCAN1CFRISTSL register is a read-only register that can be read in 16-bit units.
RSCAN1CFRISTSLL register is a read-only register that can be read in 8-bit units.

**Address:** RSCAN1CFRISTS: <RSCAN1_base> + $0248_H$

RSCAN1CFRISTSL: <RSCAN1_base> + $0248_H$

RSCAN1CFRISTSLL: <RSCAN1_base> + $0248_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CF2RXIF | CF1RXIF | CF0RXIF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.258 RSCAN1CFRISTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 | CF2RXIF | Transmit/Receive FIFO Buffer Receive Interrupt Request Status Flag<br>0: No transmit/receive FIFO buffer k receive interrupt request is present.<br>1: A transmit/receive FIFO buffer k receive interrupt request is present.<br>(k = 0 to 2) |
| 1 | CF1RXIF | |
| 0 | CF0RXIF | |

The RSCAN1CFRISTS register is cleared to $0000\ 0000_H$ in global reset mode.

### CFkRXIF Flag (k = 0 to 2)

The CFkRXIF flag is set to 1 when the CFRXIF flag in the RSCAN1CFSTSk register is set to 1 (a transmit/receive FIFO receive interrupt request is present). When the CFRXIF flag is cleared to 0, the CFkRXIF flag is cleared to 0.

## 21.16.43 RSCAN1CFTISTS — Transmit/receive FIFO Buffer Transmit Interrupt Flag Status Register

**Access:** RSCAN1CFTISTS register is a read-only register that can be read in 32-bit units.
RSCAN1CFTISTSL register is a read-only register that can be read in 16-bit units.
RSCAN1CFTISTSLL register is a read-only register that can be read in 8-bit units.

**Address:** RSCAN1CFTISTS: <RSCAN1_base> + $024C_H$

RSCAN1CFTISTSL: <RSCAN1_base> + $024C_H$

RSCAN1CFTISTSLL: <RSCAN1_base> + $024C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CF2TXIF | CF1TXIF | CF0TXIF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.259 RSCAN1CFTISTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 | CF2TXIF | Transmit/Receive FIFO Buffer Transmit Interrupt Request Status Flag<br>0: No transmit/receive FIFO buffer k transmit interrupt request is present.<br>1: A transmit/receive FIFO buffer k transmit interrupt request is present.<br>(k = 0 to 2) |
| 1 | CF1TXIF | |
| 0 | CF0TXIF | |

The RSCAN1CFTISTS register is cleared to $0000\ 0000_H$ in global reset mode.

### CFkTXIF Flag (k = 0 to 2)

The CFkTXIF flag is set to 1 when the CFTXIF flag in the RSCAN1CFSTSk register is set to 1 (a transmit/receive FIFO transmit interrupt request is present). When the CFTXIF flag is cleared to 0, the CFkTXIF flag is cleared to 0.

## 21.16.44 RSCAN1TMCp — Transmit Buffer Control Register (p = 0 to i × 16 + 15)

**Access:** RSCAN1TMCp register can be read or written in 8-bit units.

**Address:** RSCAN1TMCp: <RSCAN1_base> + $0250_H$ + ($01_H$ × p)

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TMOM | TMTAR | TMTR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W*1 | R/W*1 |

Note 1. The only effective value for writing to this bit is 1, which sets the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.260 RSCAN1TMCp Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 | TMOM | One-Shot Transmission Enable<br>0: One-shot transmission is disabled.<br>1: One-shot transmission is enabled. |
| 1 | TMTAR | Transmit Abort Request<br>0: Transmit abort is not requested.<br>1: Transmit abort is requested. |
| 0 | TMTR | Transmit Request<br>0: Transmission is not requested.<br>1: Transmission is requested. |

When the RSCAN1TMCp register meets any of the following conditions, set it to $00_H$.

- The RSCAN1TMCp register corresponds to the transmit buffer number selected by the CFTML[3:0] bits in the RSCAN1CFCCk register (p = i × 16 + the value of CFTML[3:0] bits).
- The RSCAN1TMCp register corresponds to the transmit buffer allocated to the transmit queue by the TXQDC[3:0] bits in the RSCAN1TXQCCm (m = 6) register. (p = (i × 16 + 15) to (i × 16 + 15 – the value of TXQDC[3:0] bits)).

Bits in the RSCAN1TMCp register are all cleared to 0 in channel reset mode. Modify the RSCAN1TMCp register in channel communication mode or channel halt mode.

### TMOM Bit

Setting this bit to 1 enables one-shot transmission. When transmission fails, retransmission defined in the CAN protocol is not performed.

Modify the TMOM bit when the TMTRM flag in the RSCAN1TMSTSp register is set to 0. Set the TMOM bit to 1 together with the TMTR bit.

### TMTAR Bit

Setting this bit to 1 generates a transmit abort request for the message stored in the transmit buffer. However, a message that is being transmitted or one that will be transmitted next cannot be aborted.

The TMTAR bit can be set to 1 when TMTR bit is 1.

The TMTAR bit is cleared to 0 when any of the following conditions is met, but cannot be cleared by the program writing 0 to the bit.

- Transmission has been completed.
- Transmit abort has been completed.
- An error or arbitration-lost has been detected.

If this bit becomes 0 at the same time as the program writes 1 to this bit, this bit becomes 0.

### TMTR Bit

Setting this bit to 1 transmits the message stored in the transmit buffer.

The TMTR bit is cleared to 0 when any of the following conditions is met, but cannot be cleared by the program writing 0 to the bit.

- Transmission has been completed.
- Transmit abort has been completed after the TMTAR bit was set to 1.
- An error or arbitration-lost has been detected with the TMOM bit set to 1.

Set the TMTR bit to 1 when the value of TMTRF[1:0] in the RSCAN1TMSTSp register is $00_B$.

## 21.16.45 RSCAN1TMSTSp — Transmit Buffer Status Register (p = 0 to 15)

**Access:** RSCAN1TMSTSp register can be read or written in 8-bit units.

**Address:** RSCAN1TMSTSp: <RSCAN1_base> + $02D0_H$ + ($01_H$ × p)

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | TMTARM | TMTRM | TMTRF[1:0] | | TMTSTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R |

**Table 21.261 RSCAN1TMSTSp Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 5 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 4 | TMTARM | Transmit Buffer Transmit Abort Request Status Flag<br>0: No transmit abort request is present.<br>1: A transmit abort request is present. |
| 3 | TMTRM | Transmit Buffer Transmit Request Status Flag<br>0: No transmit request is present.<br>1: A transmit request is present. |
| 2, 1 | TMTRF[1:0] | Transmit Buffer Transmit Result Status Flag<br>b2 b1<br>0 0: Transmission is in progress or no transmit request is present.<br>0 1: Transmit abort has been completed.<br>1 0: Transmission has been completed (without transmit abort request).<br>1 1: Transmission has been completed (with transmit abort request). |
| 0 | TMTSTS | Transmit Buffer Transmit Status Flag<br>0: Transmission is not in progress.<br>1: Transmission is in progress. |

The RSCAN1TMSTSp register is cleared to all 0 in channel reset mode.

### TMTARM Flag

The TMTARM flag is set to 1 when the TMTAR bit in the RSCAN1TMCp register is set to 1.

The TMTARM flag is set to 0 when the TMTAR bit in the RSCAN1TMCp register is set to 0.

### TMTRM Flag

The TMTRM flag is set to 1 when the TMTR bit in the RSCAN1TMCp register is set to 1.

The TMTRM flag is set to 0 when the TMTR bit in the RSCAN1TMCp register is set to 0.

### TMTRF[1:0] Flag

This flag indicates the result of transmission from the transmit buffer.

$00_B$: Transmission is in progress or no transmit request is present.

$01_B$: Transmission from the transmit buffer was aborted.

$10_B$: Transmission has been completed with the TMTAR bit in the RSCAN1TMCp register set to 0 (transmit abort is not requested).

$11_B$: Transmission has been completed with the TMTAR bit in the RSCAN1TMCp register set to 1 (transmit abort is requested).

Write $00_B$ to the TMTRF[1:0] flag in channel communication mode or channel halt mode. Do not write any value other than $00_B$ to this flag.

#### TMTSTS Flag

This flag is set to 1 when transmission from the transmit buffer starts, and is cleared to 0 when transmission from the transmit buffer has been completed or terminated due to a bus error or arbitration lost.

## 21.16.46 RSCAN1TMTRSTSy — Transmit Buffer Transmit Request Status Register (y = 0)

**Access:** RSCAN1TMTRSTSy register is a read-only register that can be read in 32-bit units.
RSCAN1TMTRSTSyL register is a read-only register that can be read in 16-bit units.
RSCAN1TMTRSTSyLL, RSCAN1TMTRSTSyLH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1TMTRSTSy: <RSCAN1_base> + $0350_H$ + ($04_H$ × y)

RSCAN1TMTRSTSyL: <RSCAN1_base> + $0350_H$ + ($04_H$ × y)

RSCAN1TMTRSTSyLL: <RSCAN1_base> + $0350_H$ + ($04_H$ × y),
RSCAN1TMTRSTSyLH: <RSCAN1_base> + $0351_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTRSTSp (p = y × 32 + 15 to y × 32 + 0 (y = 0)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.262 RSCAN1TMTRSTSy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. |
| 15 to 0 | TMTRSTSp | Transmit Buffer Transmit Request Status Flag p (p = y × 32 + 15 to y × 32 + 0)<br>0: No transmit request is present.<br>1: A transmit request is present. |

### TMTRSTSp Flags (p = 0 to 15)

These flags indicate the status of the TMTR bit in the RSCAN1TMCp register.

When the TMTR bit is set to 1 (transmission is requested), the corresponding TMTRSTSp flag is set to 1.

The corresponding TMTRSTSp flag is cleared to 0 when the TMTR bit is set to 0 (transmission is not requested) or in channel reset mode.

**Table 21.263** shows the bit assignment.

**Table 21.263 TMTRSTSp Bit Assignment**

| Bit | Unit Channel Number | Transmit Buffer Number |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| .<br>. | .<br>. | .<br>. |
| 15 | 0 | 15 |

## 21.16.47 RSCAN1TMTARSTSy — Transmit Buffer Transmit Abort Request Status Register (y = 0)

**Access:** RSCAN1TMTARSTSy register is a read-only register that can be read in 32-bit units.
RSCAN1TMTARSTSyL register is a read-only register that can be read in 16-bit units.
RSCAN1TMTARSTSyLL, RSCAN1TMTARSTSyLH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1TMTARSTSy: <RSCAN1_base> + $0360_H$ + ($04_H$ × y)

RSCAN1TMTARSTSyL: <RSCAN1_base> + $0360_H$ + ($04_H$ × y)

RSCAN1TMTARSTSyLL: <RSCAN1_base> + $0360_H$ + ($04_H$ × y),
RSCAN1TMTARSTSyLH: <RSCAN1_base> + $0361_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTARSTSp (p = y × 32 + 15 to y × 32 + 0 (y = 0)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.264 RSCAN1TMTARSTSy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. |
| 15 to 0 | TMTARSTSp | Transmit Buffer Transmit Abort Request Status Flag p (p = y × 32 + 15 to y × 32 + 0)<br>0: No transmit abort request is present.<br>1: A transmit abort request is present. |

### TMTARSTSp Flags (p = 0 to 15)

These flags indicate the status of the TMTAR bit in the RSCAN1TMCp register.

When the TMTAR bit is set to 1 (transmit abort is requested), the corresponding TMTARSTSp flag is set to 1.

The corresponding TMTARSTSp flag is cleared to 0 when the TMTAR bit is set to 0 (transmit abort is not requested) or in channel reset mode.

**Table 21.265** shows the bit assignment.

**Table 21.265 TMTARSTSp Bit Assignment**

| Bit | Unit Channel Number | Transmit Buffer Number |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| .<br>. | .<br>. | .<br>. |
| 15 | 0 | 15 |

## 21.16.48 RSCAN1TMTCSTSy — Transmit Buffer Transmit Complete Status Register (y = 0)

**Access:** RSCAN1TMTCSTSy register is a read-only register that can be read in 32-bit units.
RSCAN1TMTCSTSyL register is a read-only register that can be read in 16-bit units.
RSCAN1TMTCSTSyLL, RSCAN1TMTCSTSyLH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1TMTCSTSy: <RSCAN1_base> + $0370_H$ + ($04_H$ × y)

RSCAN1TMTCSTSyL: <RSCAN1_base> + $0370_H$ + ($04_H$ × y)

RSCAN1TMTCSTSyLL: <RSCAN1_base> + $0370_H$ + ($04_H$ × y),
RSCAN1TMTCSTSyLH: <RSCAN1_base> + $0371_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTCSTSp (p = y × 32 + 15 to y × 32 + 0 (y = 0)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.266 RSCAN1TMTCSTSy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. |
| 15 to 0 | TMTCSTSp | Transmit Buffer Transmit Complete Status Flag p (p = y × 32 + 15 to y × 32 + 0)<br>0: Transmission has not been completed.<br>1: Transmission has been completed. |

### TMTCSTSp Flags (p = 0 to 15)

When the TMTRF[1:0] flag in the RSCAN1TMSTSp register is set to $10_B$ (transmission has been completed (without transmit abort request)) or $11_B$ (transmission has been completed (with transmit abort request)), the corresponding TMTCSTSp flag is set to 1.

A TMTCSTSp flag is cleared to 0 when the corresponding TMTRF[1:0] flag is set to $00_B$ or in channel reset mode.

**Table 21.267** shows the bit assignment.

**Table 21.267 TMTCSTSp Bit Assignment**

| Bit | Unit Channel Number | Transmit Buffer Number |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| .<br>. | .<br>. | .<br>. |
| 15 | 0 | 15 |

## 21.16.49 RSCAN1TMTASTSy — Transmit Buffer Transmit Abort Status Register (y = 0)

**Access:** RSCAN1TMTASTSy register is a read-only register that can be read in 32-bit units.
RSCAN1TMTASTSyL register is a read-only register that can be read in 16-bit units.
RSCAN1TMTASTSyLL, RSCAN1TMTASTSyLH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1TMTASTSy: <RSCAN1_base> + $0380_H$ + ($04_H$ × y)

RSCAN1TMTASTSyL: <RSCAN1_base> + $0380_H$ + ($04_H$ × y)

RSCAN1TMTASTSyLL: <RSCAN1_base> + $0380_H$ + ($04_H$ × y),
RSCAN1TMTASTSyLH: <RSCAN1_base> + $0381_H$ + ($04_H$ × y)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMTASTSp (p = y × 32 + 15 to y × 32 + 0 (y = 0)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.268 RSCAN1TMTASTSy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. |
| 15 to 0 | TMTASTSp | Transmit Buffer Transmit Abort Status Flag p (p = y × 32 + 15 to y × 32 + 0)<br>0: Transmission is not aborted.<br>1: Transmission is aborted. |

### TMTASTSp Flags (p = 0 to 15)

When the TMTRF[1:0] flag in the RSCAN1TMSTSp register is set to $01_B$ (transmit abort has been completed), the corresponding TMTASTSp flag is set to 1.

A TMTASTSp flag is cleared to 0 when the corresponding TMTRF[1:0] flag is set to $00_B$ or in channel reset mode.

**Table 21.269** shows the bit assignment.

**Table 21.269 TMTASTSp Bit Assignment**

| Bit | Unit Channel Number | Transmit Buffer Number |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| .<br>. | .<br>. | .<br>. |
| 15 | 0 | 15 |

## 21.16.50 RSCAN1TMIECy — Transmit Buffer Interrupt Enable Configuration Register (y = 0)

**Access:** RSCAN1TMIECy register can be read or written in 32-bit units.
RSCAN1TMIECyL register can be read or written in 16-bit units.
RSCAN1TMIECyLL, RSCAN1TMIECyLH registers can be read or written in 8-bit units.

**Address:** RSCAN1TMIECy: <RSCAN1_base> + $0390_H$ + ($04_H$ × y)

RSCAN1TMIECyL: <RSCAN1_base> + $0390_H$ + ($04_H$ × y)

RSCAN1TMIECyLL: <RSCAN1_base> + $0390_H$ + ($04_H$ × y),
RSCAN1TMIECyLH: <RSCAN1_base> + $0391_H$ + ($04_H$ × y)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMIEp (p = y × 32 + 15 to y × 32 + 0 (y = 0)) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.270 RSCAN1TMIECy Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 15 to 0 | TMIEp | Transmit Buffer Interrupt Enable p (p = y × 32 + 15 to y × 32 + 0)<br>0: Transmit buffer interrupt is disabled.<br>1: Transmit buffer interrupt is enabled. |

### TMIEp Bits (p = 0 to 15)

When any of these bits is set to 1 and the corresponding transmission has been completed, a transmit buffer interrupt request is generated.

Modify these bits when the TMTRM flag in the corresponding RSCAN1TMSTSp register is 0 (no transmit request is present).

Write 0 to bits corresponding to transmit buffers linked to transmit/receive FIFO buffers or transmit buffers allocated to the transmit queue.

**Table 21.271** shows the bit assignment.

**Table 21.271 TMIEp Bit Assignment**

| Bit | Unit Channel Number | Transmit Buffer Number |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| .<br>. | .<br>. | .<br>. |
| 15 | 0 | 15 |

## 21.16.51 RSCAN1TMIDp — Transmit Buffer ID Register (p = 0 to 15)

**Access:** RSCAN1TMIDp register can be read or written in 32-bit units.
RSCAN1TMIDpL, RSCAN1TMIDpH registers can be read or written in 16-bit units.
RSCAN1TMIDpLL, RSCAN1TMIDpLH, RSCAN1TMIDpHL, RSCAN1TMIDpHH registers can be read or written in 8-bit units.

**Address:** RSCAN1TMIDp: <RSCAN1_base> + $1000_H$ + ($10_H$ × p)

RSCAN1TMIDpL: <RSCAN1_base> + $1000_H$ + ($10_H$ × p),
RSCAN1TMIDpH: <RSCAN1_base> + $1002_H$ + ($10_H$ × p)

RSCAN1TMIDpLL: <RSCAN1_base> + $1000_H$ + ($10_H$ × p),
RSCAN1TMIDpLH: <RSCAN1_base> + $1001_H$ + ($10_H$ × p),
RSCAN1TMIDpHL: <RSCAN1_base> + $1002_H$ + ($10_H$ × p),
RSCAN1TMIDpHH: <RSCAN1_base> + $1003_H$ + ($10_H$ × p)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMIDE | TMRTR | THLEN | TMID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.272 RSCAN1TMIDp Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | TMIDE | Transmit Buffer IDE<br>0: Standard ID<br>1: Extended ID |
| 30 | TMRTR | Transmit Buffer RTR<br>0: Data frame<br>1: Remote frame |
| 29 | THLEN | Transmit History Data Store Enable<br>0: Transmit history data is not stored in the buffer.<br>1: Transmit history data is stored in the buffer. |
| 28 to 0 | TMID[28:0] | Transmit Buffer ID Data<br>Set standard ID or extended ID.<br>For standard ID, write an ID to bits 10 to 0 and write 0 to bits 28 to 11. |

Modify this register when the TMTRM bit in the corresponding RSCAN1TMSTSp register is set to 0 (no transmit request is present). If this register is linked to a transmit/receive FIFO buffer, do not write data to this register. If this register is allocated to the transmit queue, only write data to a transmit buffer p (p = i × 16 + 15) for the corresponding channel.

### TMIDE Bit

This bit is used to set the ID format of the message to be transmitted from the transmit buffer.

### TMRTR Bit

This bit is used to set the data format of the message to be transmitted from the transmit buffer.

#### THLEN Bit

With this bit set to 1, the transmit history data of the message transmitted (label information and the number and type) are stored in the transmit history buffer after transmission is completed.

#### TMID[28:0] Bits

These bits are used to set the ID of the message to be transmitted from the transmit buffer.

## 21.16.52 RSCAN1TMPTRp — Transmit Buffer Pointer Register (p = 0 to 15)

**Access:** RSCAN1TMPTRp register can be read or written in 32-bit units.
RSCAN1TMPTRpH register can be read or written in 16-bit units.
RSCAN1TMPTRpHL, RSCAN1TMPTRpHH registers can be read or written in 8-bit units.

**Address:** RSCAN1TMPTRp: <RSCAN1_base> + $1004_H$ + ($10_H$ × p)

RSCAN1TMPTRpH: <RSCAN1_base> + $1006_H$ + ($10_H$ × p)

RSCAN1TMPTRpHL: <RSCAN1_base> + $1006_H$ + ($10_H$ × p),
RSCAN1TMPTRpHH: <RSCAN1_base> + $1007_H$ + ($10_H$ × p)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMDLC[3:0] | | | | — | — | — | — | TMPTR[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.273 RSCAN1TMPTRp Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | TMDLC[3:0] | Transmit Buffer DLC Data<br>b31 b30 b29 b28<br>0 0 0 0: 0 data bytes<br>0 0 0 1: 1 data byte<br>0 0 1 0: 2 data bytes<br>0 0 1 1: 3 data bytes<br>0 1 0 0: 4 data bytes<br>0 1 0 1: 5 data bytes<br>0 1 1 0: 6 data bytes<br>0 1 1 1: 7 data bytes<br>1 X X X: 8 data bytes |
| 27 to 24 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 23 to 16 | TMPTR[7:0] | Transmit Buffer Label Data<br>Set the label information to be stored in the transmit history buffer. |
| 15 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

Modify this register when the TMTRM bit in the corresponding RSCAN1TMSTSp register is set to 0 (no transmit request is present). If this register is linked to a transmit/receive FIFO buffer, do not write to this register. If this register is allocated to the transmit queue, only write to a transmit buffer p (p = i × 16 + 15) for the corresponding channel.

### TMDLC[3:0] Bits

These bits are used to set the data length of the message to be transmitted from the transmit buffer when the TMRTR bit in the RSCAN1TMIDp register is set to 0 (data frame). If the data length is set to 9 bytes or more, the transmit data is 8 bytes long.

When the TMRTR bit is set to 1 (remote frame), set the data length of messages to be requested.

**TMPTR[7:0] Bits**

When message transmission has been completed, the TMPTR[7:0] value is stored in the transmit history buffer.

## 21.16.53 RSCAN1TMDF0p — Transmit Buffer Data Field 0 Register (p = 0 to 15)

**Access:** RSCAN1TMDF0p register can be read or written in 32-bit units.
RSCAN1TMDF0pL, RSCAN1TMDF0pH registers can be read or written in 16-bit units.
RSCAN1TMDF0pLL, RSCAN1TMDF0pLH, RSCAN1TMDF0pHL, RSCAN1TMDF0pHH registers can be read or written in 8-bit units.

**Address:** RSCAN1TMDF0p: <RSCAN1_base> + $1008_H$ + ($10_H$ × p)

RSCAN1TMDF0pL: <RSCAN1_base> + $1008_H$ + ($10_H$ × p),
RSCAN1TMDF0pH: <RSCAN1_base> + $100A_H$ + ($10_H$ × p)

RSCAN1TMDF0pLL: <RSCAN1_base> + $1008_H$ + ($10_H$ × p),
RSCAN1TMDF0pLH: <RSCAN1_base> + $1009_H$ + ($10_H$ × p),
RSCAN1TMDF0pHL: <RSCAN1_base> + $100A_H$ + ($10_H$ × p),
RSCAN1TMDF0pHH: <RSCAN1_base> + $100B_H$ + ($10_H$ × p)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMDB3[7:0] | | | | | | | | TMDB2[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMDB1[7:0] | | | | | | | | TMDB0[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.274 RSCAN1TMDF0p Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | TMDB3[7:0] | Transmit Buffer Data Byte 3<br>Transmit Buffer Data Byte 2<br>Transmit Buffer Data Byte 1<br>Transmit Buffer Data Byte 0<br>Set the transmit buffer data. |
| 23 to 16 | TMDB2[7:0] | |
| 15 to 8 | TMDB1[7:0] | |
| 7 to 0 | TMDB0[7:0] | |

Modify this register when the TMTRM bit in the corresponding RSCAN1TMSTSp register is set to 0 (no transmit request is present). If this register is linked to a transmit/receive FIFO buffer, do not write to this register. If this register is allocated to the transmit queue, only write to a transmit buffer p (p = i × 16 + 15) for the corresponding channel.

## 21.16.54 RSCAN1TMDF1p — Transmit Buffer Data Field 1 Register (p = 0 to 15)

**Access:** RSCAN1TMDF1p register can be read or written in 32-bit units.
RSCAN1TMDF1pL, RSCAN1TMDF1pH registers can be read or written in 16-bit units.
RSCAN1TMDF1pLL, RSCAN1TMDF1pLH, RSCAN1TMDF1pHL, RSCAN1TMDF1pHH registers can be read or written in 8-bit units.

**Address:** RSCAN1TMDF1p: <RSCAN1_base> + $100C_H$ + ($10_H$ × p)

RSCAN1TMDF1pL: <RSCAN1_base> + $100C_H$ + ($10_H$ × p),
RSCAN1TMDF1pH: <RSCAN1_base> + $100E_H$ + ($10_H$ × p)

RSCAN1TMDF1pLL: <RSCAN1_base> + $100C_H$ + ($10_H$ × p),
RSCAN1TMDF1pLH: <RSCAN1_base> + $100D_H$ + ($10_H$ × p),
RSCAN1TMDF1pHL: <RSCAN1_base> + $100E_H$ + ($10_H$ × p),
RSCAN1TMDF1pHH: <RSCAN1_base> + $100F_H$ + ($10_H$ × p)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMDB7[7:0] | | | | | | | | TMDB6[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMDB5[7:0] | | | | | | | | TMDB4[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.275 RSCAN1TMDF1p Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | TMDB7[7:0] | Transmit Buffer Data Byte 7 |
| 23 to 16 | TMDB6[7:0] | Transmit Buffer Data Byte 6 |
| 15 to 8 | TMDB5[7:0] | Transmit Buffer Data Byte 5 |
| 7 to 0 | TMDB4[7:0] | Transmit Buffer Data Byte 4<br><br>Set the transmit buffer data. |

Modify this register when the TMTRM bit in the corresponding RSCAN1TMSTSp register is set to 0 (no transmit request is present). If this register is linked to a transmit/receive FIFO buffer, do not write to this register. If this register is allocated to the transmit queue, only write to a transmit buffer p (p = i × 16 + 15) for the corresponding channel.

## 21.16.55 RSCAN1TXQCCm — Transmit Queue Configuration and Control Register (m = 6)

**Access:** RSCAN1TXQCCm register can be read or written in 32-bit units.
RSCAN1TXQCCmL register can be read or written in 16-bit units.
RSCAN1TXQCCmLL, RSCAN1TXQCCmLH registers can be read or written in 8-bit units.

**Address:** RSCAN1TXQCCm: <RSCAN1_base> + $03A0_H$ + ($04_H$ × i)

RSCAN1TXQCCmL: <RSCAN1_base> + $03A0_H$ + ($04_H$ × i)

RSCAN1TXQCCmLL: <RSCAN1_base> + $03A0_H$ + ($04_H$ × i),
RSCAN1TXQCCmLH: <RSCAN1_base> + $03A1_H$ + ($04_H$ × i)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | TXQIM | TXQIE | TXQDC[3:0] | | | | — | — | — | — | — | — | — | TXQE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W |

**Table 21.276 RSCAN1TXQCCm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 14 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 13 | TXQIM | Transmit Queue Interrupt Source Select<br>0: When the transmit queue becomes empty upon completion of message transmission, a transmit queue interrupt request is generated.<br>1: A transmit queue interrupt request is generated each time a message has been transmitted. |
| 12 | TXQIE | Transmit Queue Interrupt Enable<br>0: Transmit queue interrupt is disabled.<br>1: Transmit queue interrupt is enabled. |
| 11 to 8 | TXQDC[3:0] | Transmit Queue Depth Configuration<br>Setting these bits to g (g = 2 to 15) makes the (g + 1)-buffer transmit queue available.<br>Setting these bits to 0 disables the transmit queue.<br>Setting these bits to 1 is prohibited. |
| 7 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | TXQE | Transmit Queue Enable<br>0: The transmit queue is not used.<br>1: The transmit queue is used. |

### TXQIM Bit

This bit is used to select a transmit queue interrupt source. Modify this bit in channel reset mode.

### TXQIE Bit

When the TXQIE bit is set to 1 and the source selected by the TXQIM bit occurs, an interrupt request is generated.

Set the TXQE bit to 0 before modifying the TXQIE bit.

#### TXQDC[3:0] Bits

These bits are used to specify the number of transmit buffers to be allocated to the transmit queues. Transmit buffers are allocated to transmit queues in descending order of buffer number, that is, from (i × 16 + 15) to (i × 16 + 0). For examples of how buffer allocation is done, see **Figure 21.47**. Modify these bits only in channel reset mode.

#### TXQE Bit

Setting this bit to 1 makes the transmit queue available. Modify this bit in channel communication mode or channel halt mode. This bit is cleared to 0 in channel reset mode.

Before setting the TXQE bit to 1, set the TXQDC[3:0] bits to $0010_B$ or more.

## 21.16.56 RSCAN1TXQSTSm — Transmit Queue Status Register (m = 6)

**Access:** RSCAN1TXQSTSm register can be read or written in 32-bit units.
RSCAN1TXQSTSmL register can be read or written in 16-bit units.
RSCAN1TXQSTSmLL register can be read or written in 8-bit units.

**Address:** RSCAN1TXQSTSm: <RSCAN1_base> + $03C0_H$ + ($04_H$ × i)

RSCAN1TXQSTSmL: <RSCAN1_base> + $03C0_H$ + ($04_H$ × i)

RSCAN1TXQSTSmLL: <RSCAN1_base> + $03C0_H$ + ($04_H$ × i)

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TXQIF | TXQFLL | TXQEMP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W*1 | R | R |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.277 RSCAN1TXQSTSm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 to 8 | Reserved | When read, the undefined value is returned. When writing, write “0”. |
| 7 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 | TXQIF | Transmit Queue Interrupt Request Flag<br>0: No transmit queue interrupt request is present.<br>1: A transmit queue interrupt request is present. |
| 1 | TXQFLL | Transmit Queue Full Status Flag<br>0: The transmit queue is not full.<br>1: The transmit queue is full. |
| 0 | TXQEMP | Transmit Queue Empty Status Flag<br>0: The transmit queue contains messages.<br>1: The transmit queue contains no message (transmit queue empty). |

### TXQIF Flag

The TXQIF flag is set to 1 when the event specified by the TXQIM bit in the RSCAN1TXQCCm register has occurred.

The TXQIF flag is cleared to 0 in channel reset mode or by writing 0 to this flag. This flag is not cleared to 0 by setting the TXQE bit in the RSCAN1TXQCCm register to 0 (the transmit queue is not used).

### TXQFLL Flag

The TXQFLL flag is set to 1 when the number of messages set for the transmit queue matches the transmit queue depth set by the TXQDC[3:0] bits in the RSCAN1TXQCCm register.

This flag is cleared to 0 in any of the following cases.

- The number of messages set for the transmit queue is smaller than the transmit queue depth set by the TXQDC[3:0] bits.
- In channel reset mode

#### TXQEMP Flag

The TXQEMP flag is cleared to 0 when even a single message is set for the transmit queue.

This flag is set to 1 in any of the following cases.

- The TXQE bit is set to 0 (the transmit queue is not used).
- The transmit queue becomes empty.
- In channel reset mode

## 21.16.57 RSCAN1TXQPCTRm — Transmit Queue Pointer Control Register (m = 6)

**Access:** RSCAN1TXQPCTRm register is a write-only register that can be written in 32-bit units.
RSCAN1TXQPCTRmL register is a write-only register that can be written in 16-bit units.
RSCAN1TXQPCTRmLL register is a write-only register that can be written in 8-bit units.

**Address:** RSCAN1TXQPCTRm: <RSCAN1_base> + $03E0_H$ + ($04_H$ × i)

RSCAN1TXQPCTRmL: <RSCAN1_base> + $03E0_H$ + ($04_H$ × i)

RSCAN1TXQPCTRmLL: <RSCAN1_base> + $03E0_H$ + ($04_H$ × i)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TXQPC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**Table 21.278 RSCAN1TXQPCTRm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing, write the value after reset. |
| 7 to 0 | TXQPC[7:0] | Transmit Queue Pointer Control<br>Writing $FF_H$ to these bits moves the write pointer of the transmit queue to the next queue buffer. |

### TXQPC[7:0] Bits

Writing $FF_H$ to the TXQPC[7:0] bits moves the write pointer to the next transmit queue buffer and generates a transmit request of the message. Write transmit messages to the RSCAN1TMIDp, RSCAN1TMPTRp, RSCAN1TMDF0p, and RSCAN1TMDF1p registers (p = 15, 31, 47, 63, 79, and 95) before writing $FF_H$ to the TXQPC[7:0] bits.

When writing $FF_H$ to these bits, make sure that the TXQE bit in the RSCAN1TXQCCm register is set to 1 (the transmit queue is used) and the TXQFLL flag in the RSCAN1TXQSTSm register is 0 (the transmit queue is not full).

## 21.16.58 RSCAN1THLCCm — Transmit History Configuration and Control Register (m = 6)

**Access:** RSCAN1THLCCm register can be read or written in 32-bit units.
RSCAN1THLCCmL register can be read or written in 16-bit units.
RSCAN1THLCCmLL, RSCAN1THLCCmLH registers can be read or written in 8-bit units.

**Address:** RSCAN1THLCCm: <RSCAN1_base> + $0400_H$ + ($04_H$ × i)

RSCAN1THLCCmL: <RSCAN1_base> + $0400_H$ + ($04_H$ × i)

RSCAN1THLCCmLL: <RSCAN1_base> + $0400_H$ + ($04_H$ × i),
RSCAN1THLCCmLH: <RSCAN1_base> + $0401_H$ + ($04_H$ × i)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | THLDTE | THLIM | THLIE | — | — | — | — | — | — | — | THLE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W |

**Table 21.279 RSCAN1THLCCm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 10 | THLDTE | Transmit History Target Buffer Select<br>0: Entry from transmit/receive FIFO buffers and transmit queue<br>1: Entry from transmit buffers, transmit/receive FIFO buffers, and transmit queue |
| 9 | THLIM | Transmit History Interrupt Source Select<br>0: When 12 sets of data have been stored in the transmit history buffer<br>1: When a single set of transmit history data has been stored |
| 8 | THLIE | Transmit History Interrupt Enable<br>0: Transmit history interrupt is disabled.<br>1: Transmit history interrupt is enabled. |
| 7 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | THLE | Transmit History Buffer Enable<br>0: Transmit history buffer is not used.<br>1: Transmit history buffer is used. |

### THLDTE Bit

When this bit is set to 0, the transmit history data of messages transmitted from transmit/receive FIFO buffers and the transmit queue is stored in the transmit history buffer. When this bit is set to 1, the transmit history data of messages transmitted from transmit buffers, transmit/receive FIFO buffers, and the transmit queue is stored in the transmit history buffer.

Modify this bit only in channel reset mode.

#### THLIM Bit

This bit is used to select a transmit history interrupt source.

Modify this bit only in channel reset mode.

#### THLIE Bit

When the THLIE bit is set to 1 and the source selected by the THLIM bit has occurred, a transmit history interrupt request is generated. Modify the THLIE bit only when the THLE bit set to 0.

#### THLE Bit

Setting this bit to 1 makes the transmit history buffer available. When data transmission from the buffer selected by the THLDTE bit has been completed, the transmit history data of transmit messages is stored in the transmit history buffer.

Modify this bit in channel communication mode or channel halt mode.

## 21.16.59 RSCAN1THLSTSm — Transmit History Status Register (m = 6)

**Access:** RSCAN1THLSTSm register can be read or written in 32-bit units.
RSCAN1THLSTSmL register can be read or written in 16-bit units.
RSCAN1THLSTSmLL register can be read or written in 8-bit units.
RSCAN1THLSTSmLH register is a read-only register that can be read in 8-bit units.

**Address:** RSCAN1THLSTSm: <RSCAN1_base> + $0420_H$ + ($04_H$ × i)

RSCAN1THLSTSmL: <RSCAN1_base> + $0420_H$ + ($04_H$ × i)

RSCAN1THLSTSmLL: <RSCAN1_base> + $0420_H$ + ($04_H$ × i),
RSCAN1THLSTSmLH: <RSCAN1_base> + $0421_H$ + ($04_H$ × i)

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | THLMC[4:0] | | | | | — | — | — | — | THLIF | THLELT | THLFLL | THLEMP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W*1 | R/W*1 | R | R |

Note 1. The only effective value for writing to this flag bit is 0, which clears the bit. Otherwise writing to the bit results in retention of its state.

**Table 21.280 RSCAN1THLSTSm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 to 8 | THLMC[4:0] | Transmit History Buffer Unread Data Counter<br>These bits indicate the number of unread data sets stored in the transmit history buffer. |
| 7 to 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 | THLIF | Transmit History Interrupt Request Flag<br>0: No transmit history interrupt request is present.<br>1: A transmit history interrupt request is present. |
| 2 | THLELT | Transmit History Buffer Overflow Flag<br>0: Transmit history buffer overflow has not occurred.<br>1: Transmit history buffer overflow has occurred. |
| 1 | THLFLL | Transmit history Buffer Full Status Flag<br>0: Transmit history buffer is not full.<br>1: Transmit history buffer is full. |
| 0 | THLEMP | Transmit History Buffer Empty Status Flag<br>0: Transmit history buffer contains unread data.<br>1: Transmit history buffer contains no unread data (buffer empty). |

### THLMC[4:0] Bits

These bits indicate the number of unread data sets stored in the transmit history buffer.

### THLIF Flag

The THLIF flag is set to 1 when the interrupt source specified with the THLIM bit in the RSCAN1THLCCm register occurs.

This flag is cleared to 0 in channel reset mode or by the program writing 0 to this flag.

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared.

When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

### THLELT Flag

The THLELT flag is set to 1 when an attempt is made to store new transmit history data while the transmit history buffer is full. In this case, the new data is discarded. This flag becomes 0 in channel reset mode or by the program writing 0 to this flag.

To clear the flags of the register to 0, the program must write 0 to the corresponding flag to be cleared.

When writing 0, using store instruction, set the bit to be set to “0” to “0”, and the bits not to be set to “0” to “1”.

### THLFLL Flag

The THLFLL flag is set to 1 when 16 data sets have been stored in the transmit history buffer, and is cleared to 0 when the number of data sets stored in the transmit history buffer has decreased to less than 16. This bit is also cleared to 0 in channel reset mode or when the THLE bit in the RSCAN1THLCCm register is set to 0 (transmit history buffer is not used).

### THLEMP Flag

The THLEMP flag is cleared to 0 when even a single set of transmit history data has been stored in the transmit history buffer.

This flag is set to 1 when all the data in the transmit history buffer has been read. This flag is also set to 1 in channel reset mode or when the THLE bit in the RSCAN1THLCCm register is set to 0 (transmit history buffer is not used).

**NOTE**

To clear THLIF or THLELT flag to 0, the program must write 0. When writing, use a store instruction to write “0” to the given flag and “1” to other flags.

## 21.16.60 RSCAN1THLACCm — Transmit History Access Register (m = 6)

**Access:** RSCAN1THLACCm register is a read-only register that can be read in 32-bit units.
RSCAN1THLACCmL register is a read-only register that can be read in 16-bit units.
RSCAN1THLACCmLL, RSCAN1THLACCmLH registers are read-only registers that can be read in 8-bit units.

**Address:** RSCAN1THLACCm: <RSCAN1_base> + $1800_H$ + ($04_H$ × i)

RSCAN1THLACCmL: <RSCAN1_base> + $1800_H$ + ($04_H$ × i)

RSCAN1THLACCmLL: <RSCAN1_base> + $1800_H$ + ($04_H$ × i),
RSCAN1THLACCmLH: <RSCAN1_base> + $1801_H$ + ($04_H$ × i)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TID[7:0] | | | | | | | | — | BN[3:0] | | | | BT[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.281 RSCAN1THLACCm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. |
| 15 to 8 | TID[7:0] | Label Data<br>The label information of stored data can be read. |
| 7 | Reserved | When read, the value after reset is returned. |
| 6 to 3 | BN[3:0] | Buffer Number Data<br>The buffer number of transmit source (transmit buffer, transmit/receive FIFO or transmit queue) can be read. |
| 2 to 0 | BT[2:0] | Buffer Type Data<br>b2 b1 b0<br>0 0 1: Transmit buffer<br>0 1 0: Transmit/receive FIFO buffer<br>1 0 0: Transmit queue |

### TID[7:0] Bits

These bits indicate the label information of transmit history data stored in the transmit history buffer.

### BN[3:0] Bits

These bits indicate the transmit source buffer number in the transmit history data stored in the transmit history buffer.

### BT[2:0] Bits

These bits indicate the type of the transmit source buffer in the transmit history data stored in the transmit history buffer.

## 21.16.61 RSCAN1THLPCTRm — Transmit History Pointer Control Register (m = 6)

**Access:** RSCAN1THLPCTRm register is a write-only register that can be written in 32-bit units.
RSCAN1THLPCTRmL register is a write-only register that can be written in 16-bit units.
RSCAN1THLPCTRmLL register is a write-only register that can be written in 8-bit units.

**Address:** RSCAN1THLPCTRm: <RSCAN1_base> + $0440_H$ + ($04_H$ × i)

RSCAN1THLPCTRmL: <RSCAN1_base> + $0440_H$ + ($04_H$ × i)

RSCAN1THLPCTRmLL: <RSCAN1_base> + $0440_H$ + ($04_H$ × i)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | THLPC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

**Table 21.282 RSCAN1THLPCTRm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing, write the value after reset. |
| 7 to 0 | THLPC[7:0] | Transmit History List Pointer Control<br>Writing $FF_H$ to these bits moves the read pointer to the next unread data in the transmit history buffer. |

### THLPC[7:0] Bits

When the THLPC[7:0] bits are set to $FF_H$, the read pointer moves to the next data in the transmit history buffer. At this time, the THLMC[4:0] (transmit history buffer unread data counter) value in the RSCAN1THLSTSm register is decremented. Write $FF_H$ to the THLPC[7:0] bits after reading from the RSCAN1THLACCm register.

When writing $FF_H$ to these bits, make sure that the THLE bit in the RSCAN1THLCCm register is set to 1 (transmit history buffer is used) and the THLEMP flag in the RSCAN1THLSTSm register is 0.

## 21.16.62 RSCAN1GTSTCFG — Global Test Configuration Register

**Access:** RSCAN1GTSTCFG register can be read or written in 32-bit units.
RSCAN1GTSTCFGH register can be read or written in 16-bit units.
RSCAN1GTSTCFGHL register can be read or written in 8-bit units.

**Address:** RSCAN1GTSTCFG: <RSCAN1_base> + $0468_H$

RSCAN1GTSTCFGH: <RSCAN1_base> + $046A_H$

RSCAN1GTSTCFGHL: <RSCAN1_base> + $046A_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | RTMPS[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.283 RSCAN1GTSTCFG Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 22 to 16 | RTMPS[6:0] | RAM Test Page Configuration<br>Set a value within a range of page 0 ($00_H$) to page 56 ($38_H$). |
| 15 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

Modify the RSCAN1GTSTCFG register only in global test mode.

### RTMPS[6:0] Bits

These bits are used to set the RAM test target page number for RAM test. Set a value in the range of $00_H$ to $38_H$, inclusive.

## 21.16.63 RSCAN1GTSTCTR — Global Test Control Register

**Access:** RSCAN1GTSTCTR register can be read or written in 32-bit units.
RSCAN1GTSTCTRL register can be read or written in 16-bit units.
RSCAN1GTSTCTRLL register can be read or written in 8-bit units.

**Address:** RSCAN1GTSTCTR: <RSCAN1_base> + $046C_H$

RSCAN1GTSTCTRL: <RSCAN1_base> + $046C_H$

RSCAN1GTSTCTRLL: <RSCAN1_base> + $046C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | RTME | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R |

**Table 21.284 RSCAN1GTSTCTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 | RTME | RAM Test Enable<br>0: RAM test is disabled.<br>1: RAM test is enabled. |
| 1, 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

### RTME Bit

Setting this bit to 1 enables the RAM test. Modify this bit only in global test mode.

1. Set the GMDC[1:0] bits in the RSCAN1GCTR register to $10_B$ (Global test mode).
2. Set the RTME bit to 1.
3. Check that the RTME bit is set to 1.

## 21.16.64 RSCAN1GLOCKK — Global Lock Key Register

**Access:** RSCAN1GLOCKK register is a write-only register that can be written in 32-bit units.
RSCAN1GLOCKKL register is a write-only register that can be written in 16-bit units.

**Address:** RSCAN1GLOCKK: <RSCAN1_base> + $047C_H$

RSCAN1GLOCKKL: <RSCAN1_base> + $047C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LOCK[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 | W*1 |

Note 1. Writing to these bits is effective only when the RS-CAN module is in global test mode.

**Table 21.285 RSCAN1GLOCKK Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When writing, write the value after reset. |
| 15 to 0 | LOCK[15:0] | Lock Key<br>These bits are key bits to release protection of test mode. |

The RSCAN1GLOCKK register releases protection of special test bits and is write only.

For the protection release data, see **Section 21.22.4.2, Procedure for Releasing the Protection**.

### LOCK[15:0] Bits

Writing the protection release data to the LOCK[15:0] bits in succession enables writing 1 to the RTME bit in the RSCAN1GTSTCTR register.

After the protection has been released, writing to the I/O register area (<RSCAN1_base> + $0000_H$ to <RSCAN1_base> + $04FF_H$) of the CAN (except the RAM) enables the protection again.

Reading from the I/O register area of the CAN or reading from/wring to other areas does not enable the protection.

## 21.16.65 RSCAN1RPGACCr — RAM Test Page Access Register (r = 0 to 63)

**Access:** RSCAN1RPGACCr register can be read or written in 32-bit units.
RSCAN1RPGACCrL, RSCAN1RPGACCrH registers can be read or written in 16-bit units.
RSCAN1RPGACCrLL, RSCAN1RPGACCrLH, RSCAN1RPGACCrHL, RSCAN1RPGACCrHH registers can be read or written in 8-bit units.

**Address:** RSCAN1RPGACCr: <RSCAN1_base> + $1900_H$ + ($04_H$ × r)

RSCAN1RPGACCrL: <RSCAN1_base> + $1900_H$ + ($04_H$ × r),
RSCAN1RPGACCrH: <RSCAN1_base> + $1902_H$ + ($04_H$ × r)

RSCAN1RPGACCrLL: <RSCAN1_base> + $1900_H$ + ($04_H$ × r),
RSCAN1RPGACCrLH: <RSCAN1_base> + $1901_H$ + ($04_H$ × r),
RSCAN1RPGACCrHL: <RSCAN1_base> + $1902_H$ + ($04_H$ × r),
RSCAN1RPGACCrHH: <RSCAN1_base> + $1903_H$ + ($04_H$ × r)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDTA[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDTA[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.286 RSCAN1RPGACCr Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | RDTA [31:0] | RAM Data Test Access<br>Data can be read and written in RSCAN RAM. |

Modify the RSCAN1RPGACCr register in global test mode with the RTME bit in the RSCAN1GTSTCTR register set to 1 (RAM test is enabled).

The RSCAN1RPGACCr register is readable and writable when the RTME bit is set to 1.

## 21.17 Interrupt Sources

The RS-CAN module has 5 interrupts that are grouped into global interrupts and channel interrupts.

- Global interrupts (2 sources/unit)
  Receive FIFO interrupt (1 source/unit)
  Global error interrupt (1 source/unit)
- Channel interrupts (3 sources/channel):
  CANm transmit interrupt (m = 6)
  - CANm transmit complete interrupt
  - CANm transmit abort interrupt
  - CANm transmit/receive FIFO transmit complete interrupt (in transmit mode)
  - CANm transmit history interrupt
  - CANm transmit queue Interrupt

  CANm transmit/receive FIFO receive complete interrupt (in receive mode)
  CANm error interrupt

When an interrupt request is generated, the corresponding interrupt request flag is set to 1 (interrupt request present). In that case, when the interrupt enable bit is set to 1 (enabling interrupts), an interrupt request is output from the RS-CAN module. (Generation of interrupts also depends on the interrupt control register settings of the interrupt controller.)

Setting the interrupt request flag to 0 (no interrupt request present) or setting the interrupt enable bit to 0 (disabling interrupts) clears the current interrupt request. The current interrupt request is still output until the interrupt request flag is cleared.

**Table 21.287** lists the CAN interrupt sources. **Figure 21.40** shows the CAN global interrupt block diagram. **Figure 21.41** shows the CAN channel interrupt block diagram.

**Table 21.287 List of CAN Interrupt Sources**

| Interrupt Source | | | Corresponding Interrupt Request Flag | Corresponding Interrupt Enable Bit |
|---|---|---|---|---|
| Global interrupts | Receive FIFO | Receive FIFO 0 | RFIF in the RSCAN1RFSTS0 register | RFIE in the RSCAN1RFCC0 register |
| | | Receive FIFO 1 | RFIF in the RSCAN1RFSTS1 register | RFIE in the RSCAN1RFCC1 register |
| | | Receive FIFO 2 | RFIF in the RSCAN1RFSTS2 register | RFIE in the RSCAN1RFCC2 register |
| | | Receive FIFO 3 | RFIF in the RSCAN1RFSTS3 register | RFIE in the RSCAN1RFCC3 register |
| | | Receive FIFO 4 | RFIF in the RSCAN1RFSTS4 register | RFIE in the RSCAN1RFCC4 register |
| | | Receive FIFO 5 | RFIF in the RSCAN1RFSTS5 register | RFIE in the RSCAN1RFCC5 register |
| | | Receive FIFO 6 | RFIF in the RSCAN1RFSTS6 register | RFIE in the RSCAN1RFCC6 register |
| | | Receive FIFO 7 | RFIF in the RSCAN1RFSTS7 register | RFIE in the RSCAN1RFCC7 register |
| | Global error | | • DEF in the RSCAN1GERFL register<br>• MES in the RSCAN1GERFL register<br>• THLES in the RSCAN1GERFL register | • DEIE in the RSCAN1GCTR register<br>• MEIE in the RSCAN1GCTR register<br>• THLEIE in the RSCAN1GCTR register |
| Channel interrupts (m = 6) | CANm transmit | CANm transmit complete | TMTRF[1:0] in the RSCAN1TMSTSp register | TMIEp in the RSCAN1TMIECy register |
| | | CANm transmit abort | TMTRF[1:0] in the RSCAN1TMSTSp register | TAIE in the RSCAN1CmCTR register |
| | | CANm transmit/receive FIFO transmit complete | CFTXIF in the RSCAN1CFSTSk register | CFTXIE in the RSCAN1CFCCk register |
| | | CANm transmit queue | TXQIF in the RSCAN1TXQSTSm register | TXQIE in the RSCAN1TXQCCm register |
| | | CANm transmit history | THLIF in the RSCAN1THLSTSm register | THLIE in the RSCAN1THLCCm register |
| | CANm transmit/receive FIFO receive complete | | CFRXIF in the RSCAN1CFSTSk register | CFRXIE in the RSCAN1CFCCk register |
| | CANm error | | • BEF in the RSCAN1CmERFL register<br>• ALF in the RSCAN1CmERFL register<br>• BLF in the RSCAN1CmERFL register<br>• OVLF in the RSCAN1CmERFL register<br>• BORF in the RSCAN1CmERFL register<br>• BOEF in the RSCAN1CmERFL register<br>• EPF in the RSCAN1CmERFL register<br>• EWF in the RSCAN1CmERFL register | • BEIE in the RSCAN1CmCTR register<br>• ALIE in the RSCAN1CmCTR register<br>• BLIE in the RSCAN1CmCTR register<br>• OLIE in the RSCAN1CmCTR register<br>• BORIE in the RSCAN1CmCTR register<br>• BOEIE in the RSCAN1CmCTR register<br>• EPIE in the RSCAN1CmCTR register<br>• EWIE in the RSCAN1CmCTR register |

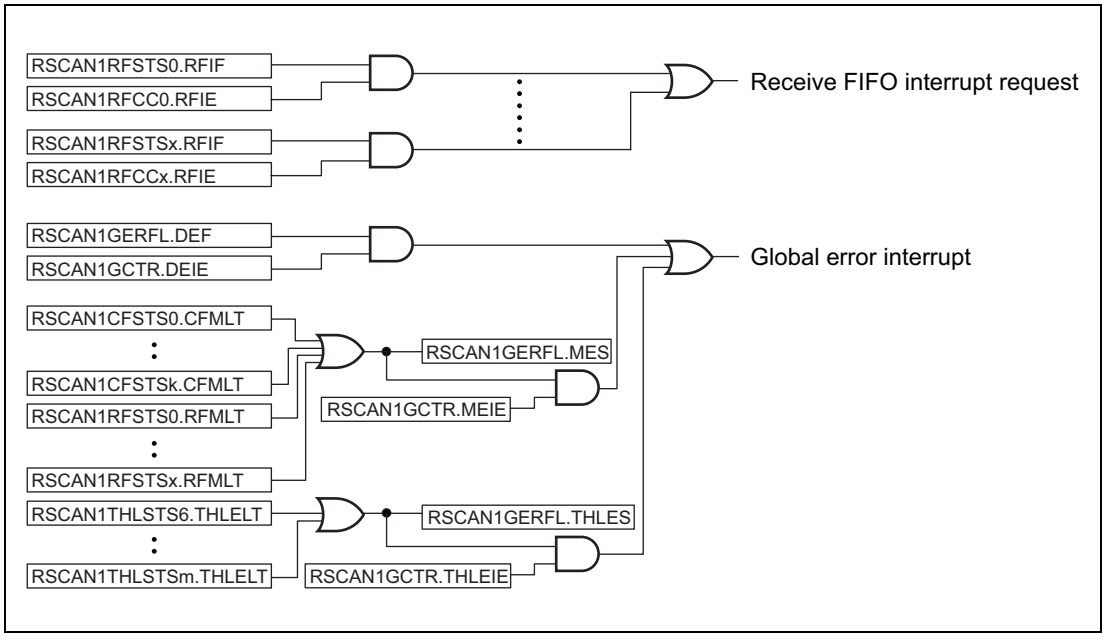



**Figure 21.40 CAN Global Interrupt Block Diagram**

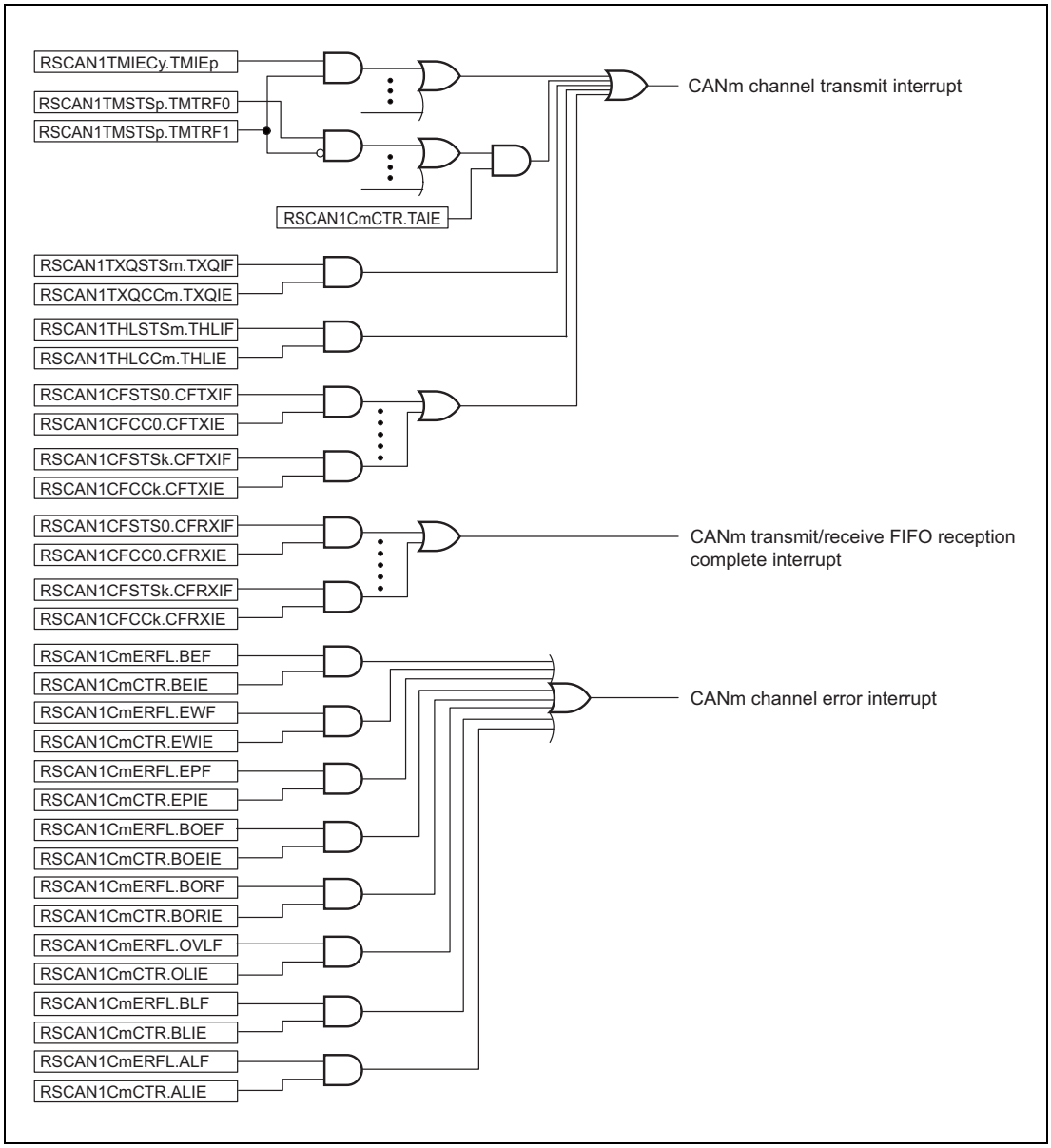



**Figure 21.41 CAN Channel Interrupt Block Diagram**

# 21.18 CAN Modes

The RS-CAN module has four global modes to control the entire RS-CAN module status and four channel modes to control individual channel status. Details of global modes are described in **Section 21.18.1, Global Modes**, and details of channel modes are described in **Section 21.18.2, Channel Modes**.

- Global stop mode: Stops the clocks of the entire module to achieve low power consumption.
- Global reset mode: Performs initial settings for the entire module.
- Global test mode: Performs test settings and performs the RAM test.
- Global operating mode: Makes the entire module operable.
- Channel stop mode: Stops the channel clock.
- Channel reset mode: Performs initial settings for the channels.
- Channel halt mode: Stops CAN communication and allows channel testing.
- Channel communication mode: Performs CAN communication.

## 21.18.1 Global Modes

**Figure 21.42** shows the transitions of global modes.

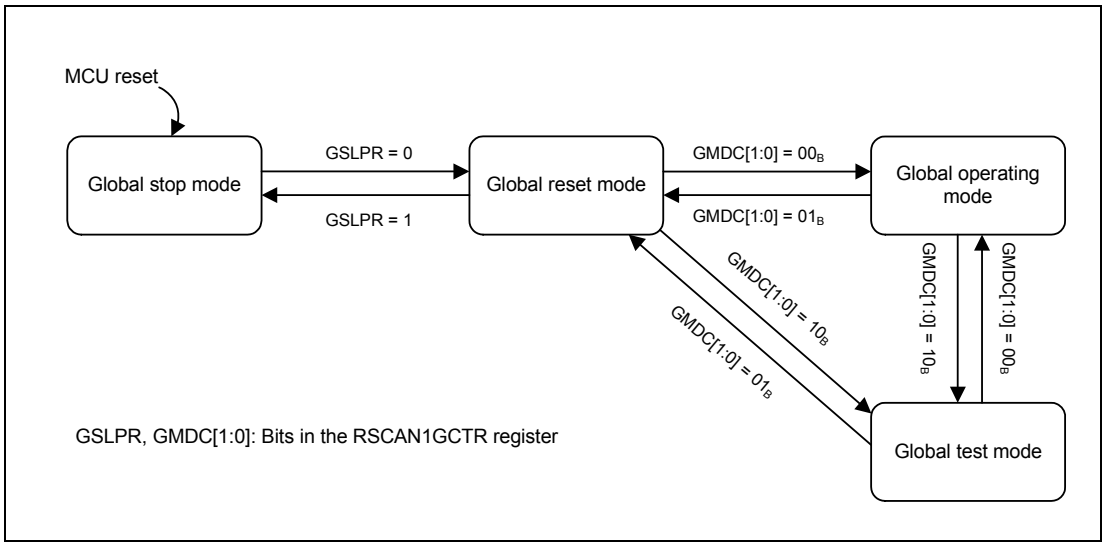



**Figure 21.42 Transitions of Global Modes**

In some cases, global mode transitions also force channel mode transitions. **Table 21.288** shows the channel mode transitions depending on the global mode setting dictated by the GMDC[1:0] bits and the GSLPR bit.

**Table 21.288 Transitions of Channel Modes Depending on Global Mode Setting (GMDC[1:0] and GSLPR Bits)**

| Channel Mode before Setting | Channel Mode after Setting | | | |
|---|---|---|---|---|
| | GMDC[1:0] = $00_B$ GSLPR = 0 (Global Operation) | GMDC[1:0] = $10_B$ GSLPR = 0 (Global Test) | GMDC[1:0] = $01_B$ GSLPR = 0 (Global Reset) | GMDC[1:0] = $01_B$ GSLPR = 1 (Global Stop) |
| Channel communication | Channel communication | Channel halt | Channel reset | Transition prohibited |
| Channel halt | Channel halt | Channel halt | Channel reset | Transition prohibited |
| Channel reset | Channel reset | Channel reset | Channel reset | Channel stop |
| Channel stop | Channel stop | Channel stop | Channel stop | Channel stop |

**Note:** GMDC[1:0], GSLPR: Bits in the RSCAN1GCTR register

**Table 21.289** shows the global mode transition time.

**Table 21.289 Global Mode Transition Time**

| Mode before Transition | Mode after Transition | Maximum Transition Time |
|---|---|---|
| Global stop | Global reset | Three pclk cycles |
| Global reset | Global stop | Three pclk cycles |
| Global reset | Global test | Ten pclk cycles |
| Global reset | Global operating | Ten pclk cycles |
| Global test | Global reset | Three pclk cycles |
| Global test | Global operating | Three pclk cycles |
| Global operating | Global reset | Three pclk cycles |
| Global operating | Global test | Two CAN frames*1 |

Note 1. CAN frame time of the lowest communication speed of the channels in use

#### 21.18.1.1 Global Stop Mode

In global stop mode, clocks of the CAN do not run and therefore power consumption is reduced. CAN registers can be read, but writing data to them is prohibited. Register values are retained. Only the clock used by the CPU for writing to the GSLPR bit runs in this mode.

After the MCU is reset, the CAN module transitions to global stop mode. Setting the GSLPR bit in the RSCAN1GCTR register to 1 (in global stop mode) in global reset mode sets the CSLPR bit in each of the RSCAN1CmCTR register to 1 (channel stop mode). If all channels are forced to transition to channel stop mode, the CAN module transitions to global stop mode. The GSLPR bit should not be modified in global operating mode or global test mode.

#### 21.18.1.2 Global Reset Mode

In global reset mode, RS-CAN module settings are performed. When the RS-CAN module transitions to global reset mode, some registers are initialized. **Table 21.292** and **Table 21.293** list the registers to be initialized.

Setting the GMDC[1:0] bits in the RSCAN1GCTR register to $01_B$ sets the CHMDC[1:0] bits in each of the RSCAN1CmCTR registers (m = 6) to $01_B$ (channel reset mode). If all channels are forced to transition to channel reset mode, the CAN module transitions to global reset mode. Channels that are already in channel reset mode or channel stop mode do not transition (because the CHMDC[1:0] bits have already been set to $01_B$).

#### 21.18.1.3 Global Test Mode

In global test mode, settings for test-related registers are performed. When the CAN module transitions to global test mode, all CAN communications are disabled.

Setting the GMDC[1:0] bits in the RSCAN1GCTR register to $10_B$ sets the CHMDC[1:0] bits in each of the RSCAN1CmCTR register to $10_B$ (channel halt mode). If all channels are forced to transition to channel halt mode, the CAN module transitions to global test mode. Channels that are in channel stop mode, channel reset mode, or channel halt mode do not transition.

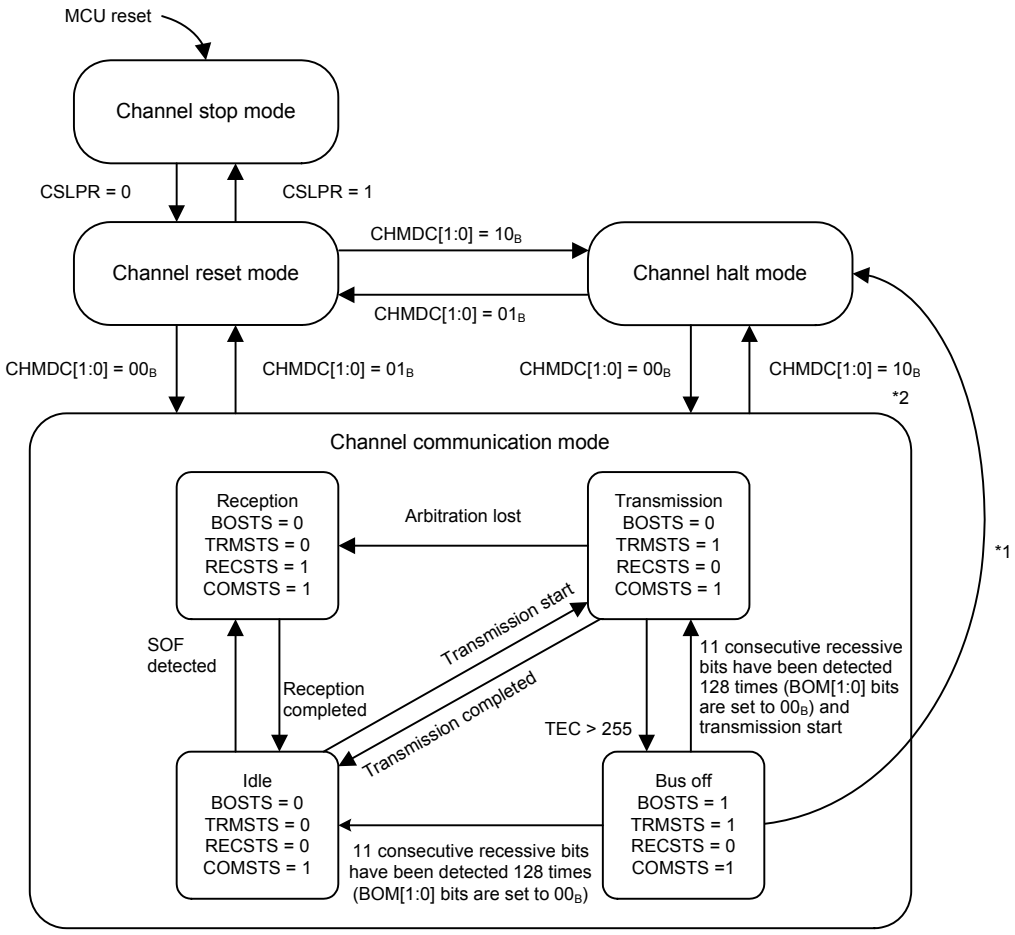
#### 21.18.1.4 Global Operating Mode

The RS-CAN module operates in global operating mode.

When the GMDC[1:0] bits in the RSCAN1GCTR register are set to $00_B$, the RS-CAN module transitions to global operating mode.

## 21.18.2 Channel Modes

**Figure 21.43** shows a channel mode state transition chart. **Table 21.290** shows the channel mode transition time.




**Note:** CHMDC[1:0], CSLPR, BOM[1:0]: Bits in the RSCAN1CmCTR register (m = 6)
BOSTS, TRMSTS, RECSTS, COMSTS: Bits in the RSCAN1CmSTS register

Note 1. Timing of transition from bus off state to channel halt mode
- When BOM[1:0] = $01_B$: Transition to channel halt mode when TEC exceeds 255
- When BOM[1:0] = $10_B$: Transition to channel halt mode when 11 consecutive recessive bits have been detected 128 times
- When BOM[1:0] = $11_B$: Transition to channel halt mode when the CHMDC[1:0] bits are set to $10_B$

Note 2. While the CAN bus is locked at the dominant level, transition to channel halt mode is not made. In that case, enter channel reset mode.

**Figure 21.43 Channel Mode State Transition Chart**

**Table 21.290 Channel Mode Transition Time (1/2)**

| Mode before Transition | Mode after Transition | Maximum Transition Time |
|---|---|---|
| Channel stop | Channel reset | Three pclk cycles |
| Channel reset | Channel stop | Three pclk cycles |
| Channel reset | Channel halt | Three CANm bit times |
| Channel reset | Channel communication | Two CANm bit times |

Table 21.290 Channel Mode Transition Time (2/2)

| Mode before Transition | Mode after Transition | Maximum Transition Time |
|---|---|---|
| Channel halt | Channel reset | Three pclk cycles |
| Channel halt | Channel communication | Three CANm bit times |
| Channel communication | Channel reset | Three pclk cycles |
| Channel communication | Channel halt | Two CANm frames |

### 21.18.2.1 Channel Stop Mode

In channel stop mode, clocks are not supplied to channels and therefore power consumption is reduced. CAN registers can be read, but writing data to them is prohibited. Register values are retained.

Each channel enters channel stop mode after the MCU is reset. Channels also transition to channel stop mode when the CSLPR bit in the RSCAN1CmCTR register (m = 6) is set to 1 (channel stop mode) in channel reset mode. The CSLPR bit should not be modified in channel communication mode and channel halt mode.

### 21.18.2.2 Channel Reset Mode

In channel reset mode, channel settings are performed. When a channel transitions to channel reset mode, some channel-related registers are initialized. **Table 21.292** lists the registers to be initialized.

When the CHMDC[1:0] bits in the RSCAN1CmCTR register are set to $01_B$ (channel reset mode) during CAN communication, communication is terminated before it is completed and the channel transitions to channel reset mode. **Table 21.291** shows the operation when the CHMDC[1:0] bits are set to $01_B$ (channel reset mode) during CAN communication.

### 21.18.2.3 Channel Halt Mode

In channel halt mode, settings for test-related registers of channels are performed. When a channel transitions to channel halt mode, CAN communication of the channel stops.

**Table 21.291** shows operation when the CHMDC[1:0] bits are set to $10_B$ (channel halt mode) during CAN communication.

Table 21.291 Operation a Channel Transitions to Channel Reset Mode/Channel Halt Mode

| Mode | During Reception | During Transmission | Bus Off State |
|---|---|---|---|
| Channel reset (CHMDC[1:0] = $01_B$) | Transitions to channel reset mode before reception is completed.*1 | Transitions to channel reset mode before transmission is completed.*1 | Transitions to channel reset mode before bus off recovery. |
| Channel halt*3 (CHMDC[1:0] = $10_B$) | Transitions to channel halt mode after reception is completed.*2 | Transitions to channel halt mode after transmission is completed. | [When BOM[1:0] = $00_B$] Transitions to channel halt mode (CHMDC[1:0] = $10_B$) only after bus off recovery. [When BOM[1:0] = $01_B$] Transitions to channel halt mode automatically when the condition for transition to bus off state is met. [When BOM[1:0] = $10_B$] Transitions to channel halt mode automatically after bus off recovery. [When BOM[1:0] = $11_B$] Transitions to channel halt mode immediately after the CHMDC[1:0] bits are set to $10_B$ before bus off recovery. |

Note 1. To allow transition to channel reset mode after communication is completed, set the CHMDC[1:0] bits to $10_B$ and confirm that communication has been completed and transition to channel halt mode has been made, and then set the CHMDC[1:0] bits to $01_B$.

Note 2. While the CAN bus is locked at the dominant level, transition to channel halt mode is not made. In that case, enter channel reset mode. The CAN bus status can be confirmed with the BLF flag of the RSCAN1CmERFL register that becomes 1 when dominant lock is detected.

Note 3. When the transition from channel reset mode to channel halt mode is to be made, set the RSCAN1CmCFG register in channel reset mode and then shift to channel halt mode.

### 21.18.2.4 Channel Communication Mode

In channel communication mode, CAN communication is performed. Each channel has the following communication states during CAN communication.

- Idle: Neither reception nor transmission is in progress.
- Reception: Receiving a message sent from another node.
- Transmission: Transmitting a message.
- Bus off: Isolated from CAN communication.

When the CHMDC[1:0] bits in the RSCAN1CmCTR register are set to $00_B$, the channel transitions to channel communication mode. After that, once 11 consecutive recessive bits have been detected, the COMSTS flag in the RSCAN1CmSTS register (m = 6) is set to 1 (communication is ready) and transmission and reception are enabled on the CAN network as an active node. At this time, transmission and reception of messages can be started.

### 21.18.2.5 Bus Off State

A channel transitions to the bus off state according to the transmit/receive error counter increment/ decrement rules of the CAN specifications.

The conditions for returning from the bus off state are determined by the BOM[1:0] bits in the RSCAN1CmCTR register.

- When BOM[1:0] = $00_B$:
  Bus off recovery is compliant with the CAN specifications. After 11 consecutive recessive bits have been detected 128 times, a channel returns from the bus off state to the CAN communication ready state (error active state). At that time, the TEC[7:0] and REC[7:0] bits in the RSCAN1CmSTS register are initialized to $00_H$, the BORF flag in the RSCAN1CmERFL register is set to 1 (bus off recovery is detected), and a bus off recovery interrupt request is generated. When the CHMDC[1:0] bits in the RSCAN1CmCTR register are set to $10_B$ (channel halt mode) in the bus off state, the channel transitions to channel halt mode after bus off recovery has been completed (11 consecutive recessive bits have been detected 128 times).
- When BOM[1:0] = $01_B$:
  When a channel transitions to the bus off state, the CHMDC[1:0] bits are set to $10_B$ and the channel transitions to channel halt mode. At that time, the TEC[7:0] and REC[7:0] bits are initialized to $00_H$, but the BORF flag is not set to 1. Also, a bus off recovery interrupt request is not generated.
- When BOM[1:0] = $10_B$:
  When a channel has transitioned to the bus off state, the CHMDC[1:0] bits are set to $10_B$. After bus off recovery has been completed (11 consecutive recessive bits have been detected 128 times), the channel transitions to channel halt mode. At that time, the TEC[7:0] and REC[7:0] bits are initialized to $00_H$, the BORF flag is set to 1, and a bus off recovery interrupt request is generated.

- When BOM[1:0] = $11_B$:
  When the CHMDC[1:0] bits are set to $10_B$ in the bus off state, the channel transitions to channel halt mode before bus off recovery is completed. At that time, the TEC[7:0] and REC[7:0] bits are initialized to $00_H$, but the BORF flag is not set to 1. Also, a bus off recovery interrupt is not generated.
  However, the BORF flag becomes 1 and a bus off recovery interrupt request is generated if a CAN module transitions to error active state (by detecting 128 times of 11 consecutive recessive bits) before CHMDC[1:0] bits are set to $10_B$.

If the RS-CAN module causes the channel to transition to channel halt mode simultaneously with a program write to the CHMDC[1:0] bits, the program write takes precedence. An automatic transition to channel halt mode when the BOM[1:0] bits are set to $01_B$ or $10_B$ is made only when the CHMDC[1:0] bits are $00_B$ (channel communication mode).

Furthermore, setting the RTBO bit in the RSCAN1CmCTR register to 1 allows a forced return from the bus off state. As soon as the RTBO bit is set to 1, the state changes to the error active state. After 11 consecutive recessive bits have been detected, the CAN module becomes ready for communication. In this case, the BORF flag is not set to 1 and the TEC[7:0] and REC[7:0] bits are initialized to $00_H$. Write 1 to the RTBO bit only when the BOM[1:0] value is $00_B$. Writing the RTBO bit to 1 in a state other than the bus off state is ignored, and the RTBO bit is immediately set to 0.

**Table 21.292 Registers Initialized in Global Reset Mode or Channel Reset Mode**

| Register | Bit / Flag |
|---|---|
| RSCAN1CmCTR register | CTMS[1:0], CTME, CHMDC[1:0] |
| RSCAN1CmSTS register | CHLTSTS, EPSTS, BOSTS, TRMSTS, RECSTS, COMSTS, REC[7:0], TEC[7:0] |
| RSCAN1CmERFL register | CRCREG[14:0], ADERR, B0ERR, B1ERR, CERR, AERR, FERR, SERR, ALF, BLF, OVLF, BORF, BOEF, EPF, EWF, BEF |
| RSCAN1CFCCk register | When transmit/receive FIFO buffer is in transmit mode: CFE |
| RSCAN1CFSTSk register | When transmit/receive FIFO buffer is in transmit mode: CFMC[7:0], CFFLL, CFEMP, CFMLT, CFRXIF, CFTXIF |
| RSCAN1CFTISTS register | CFkTXIF |
| RSCAN1TMCp register | TMOM, TMTAR, TMTR |
| RSCAN1TMSTSp register | TMTARM, TMTRM, TMTRF[1:0], TMTSTS |
| RSCAN1TMTRSTSy register | TMTRSTSp (Bits of corresponding channel are initialized in channel reset mode.) |
| RSCAN1TMTARSTSy register | TMTARSTSp (Bits of corresponding channel are initialized in channel reset mode.) |
| RSCAN1TMTCSTSy register | TMTCSTSp (Bits of corresponding channel are initialized in channel reset mode.) |
| RSCAN1TMTASTSy register | TMTASTSp (Bits of corresponding channel are initialized in channel reset mode.) |
| RSCAN1TXQCCm register | TXQE |
| RSCAN1TXQSTSm register | TXQIF, TXQFLL, TXQEMP |
| RSCAN1THLCCm register | THLE |
| RSCAN1THLSTSm register | THLMC[4:0], THLIF, THLELT, THLFLL, THLEMP |
| RSCAN1GTINTSTS0 register | TSIFi, TAIFi, TQIFi, CFTIFi, THIFi (i = 0) |

Table 21.293 Registers Initialized Only in Global Reset Mode

| Register | Bit / Flag |
|---|---|
| RSCAN1GSTS register | GHLTSTS |
| RSCAN1GERFL register | THLES, MES, DEF |
| RSCAN1GTSC register | TS[15:0] |
| RSCAN1RMNDy register | RMNSq |
| RSCAN1RFCCx register | RFE |
| RSCAN1RFSTSx register | RFMC[7:0], RFIF, RFMLT, RFFLL, RFEMP |
| RSCAN1CFCCk register | When transmit/receive FIFO buffer is in receive mode: CFE |
| RSCAN1CFSTSk register | When transmit/receive FIFO buffer is in receive mode: CFMC[7:0], CFFLL, CFEMP, CFTXIF, CFRXIF, CFMLT |
| RSCAN1FESTS register | CFkEMP, RFxEMP |
| RSCAN1FFSTS register | CFkFLL, RFxFLL |
| RSCAN1FMSTS register | CFkMLT, RFxMLT |
| RSCAN1RFISTS register | RFxIF |
| RSCAN1CFRISTS register | CFkRXIF |
| RSCAN1GTSTCFG register | RTMPS[6:0], C0ICBCE, C1ICBCE, C2ICBCE, C3ICBCE, C4ICBCE, C5ICBCE |
| RSCAN1GTSTCTR register | RTME, ICBCTME |

# 21.19 Reception Function

There are two reception types.

- Reception by receive buffers:
  Zero to 16 receive buffers can be shared by all channels. Since messages stored in receive buffers are overwritten at each reception, the latest receive data can always be read.
- Reception by receive FIFO buffers and transmit/receive FIFO buffers (receive mode):
  Eight receive FIFO buffers can be shared by all channels and three dedicated transmit/receive FIFO buffers are provided for each channel. Messages of up to the number of buffer stages specified with the RFDC[2:0] and CFDC[2:0] bits can be stored in FIFO buffers and can be read sequentially from the oldest.

## 21.19.1 Data Processing Using the Receive Rule Table

Data processing using the receive rule table allows dispatching of selected messages to the specified buffer. Data processing includes acceptance filter processing, DLC filter processing, routing processing, label addition processing, and mirror function processing.

Up to 128 receive rules can be registered per channel and up to (64 × number of channels) total receive rules can be registered in the entire module (up to 64 receive rules can be registered in this module that has one channel.) Set receive rules for each channel. Receive rules cannot be shared with other channels. If receive rules are not set, no messages can be received. **Figure 21.44** illustrates how receive rules are registered.

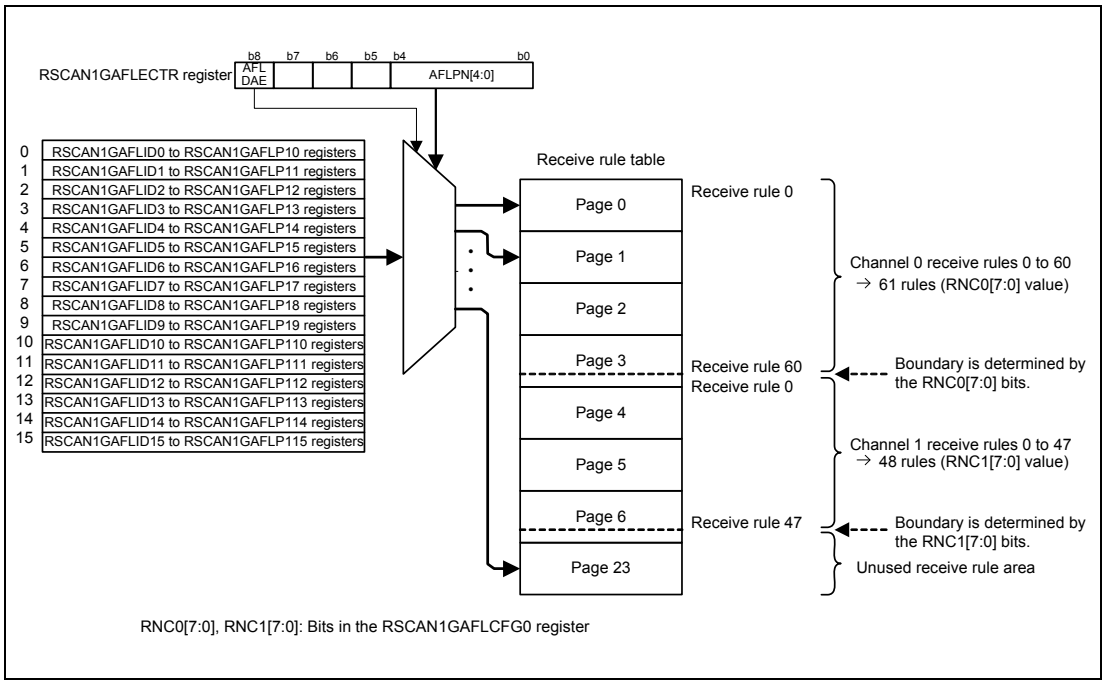



**Figure 21.44 Entry of Receive Rules (for Setting Channel 0 and 1)**

**CAUTION**

Receive rules for each channel must be set in contiguous blocks.
Channel 1 rules and channel 0 rules must be set separately.

Each receive rule consists of 16 bytes in the RSCAN1GAFLIDj, RSCAN1GAFLMj, RSCAN1GAFLP0j, and RSCAN1GAFLP1j registers (j = 0 to 15). The RSCAN1GAFLIDj register is used to set GAFLID, GAFLIDE bit, GAFLRTR bit, and the mirror function, the RSCAN1GAFLMj register is used to set mask, the RSCAN1GAFLP0j register is used to set label information to be added, DLC value, and storage receive buffer, and the RSCAN1GAFLP1j register is used to set storage FIFO buffer. Up to 16 receive rules can be set per page.

### 21.19.1.1 Acceptance Filter Processing

In the acceptance filter processing, the ID data, IDE bit, and RTR bit in a received message are compared with the ID data, IDE bit, and RTR bit set in the receive rule of the corresponding channel. When all these bits match, the message passes through the acceptance filter processing. The ID data, IDE bit, and RTR bit in the received message which correspond to the bits set to 0 (bits are not compared) in the RSCAN1GAFLMj register are not compared and are regarded as matched.

Check begins with the receive rule of the minimum number for the corresponding channel. When all the bits to be compared in a received message match the bits set in the receive rule or when all the receive rules are compared without any match, filter processing stops. If there is no matching receive rule, the received message is not stored in the receive buffer or FIFO buffer.

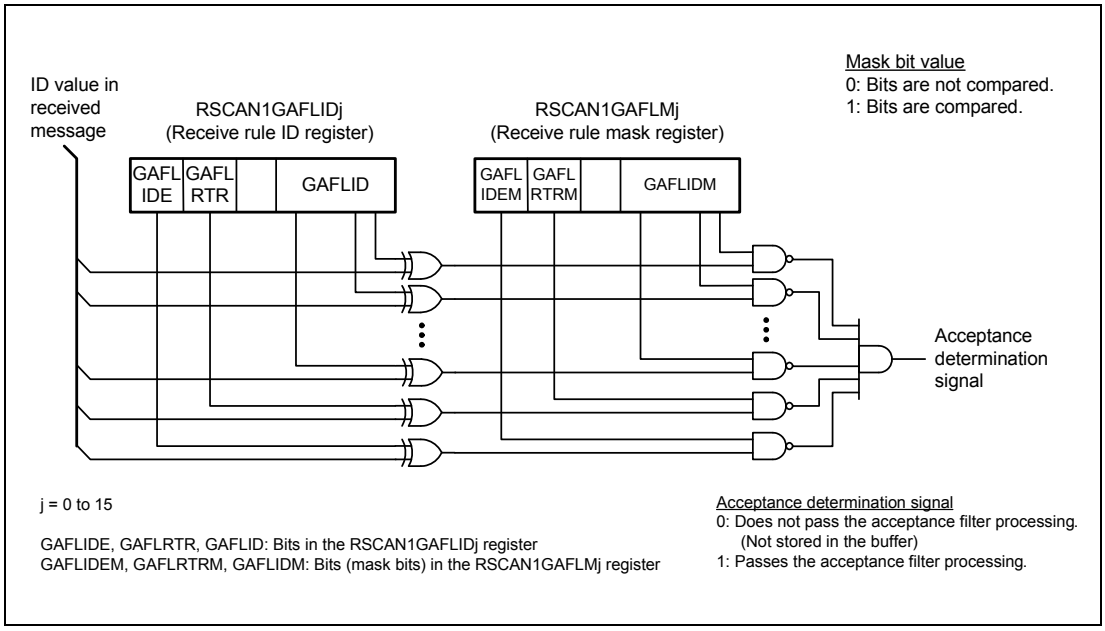



**Figure 21.45 Acceptance Filter Function**

#### 21.19.1.2 DLC Filter Processing

When the DCE bit in the RSCAN1GCFG register is set to 1 (DLC check is enabled), DLC filter processing is added to messages that passed through the acceptance filter processing. When the DLC value in a message is equal to or larger than the DLC value set in the receive rule, the message passes through the DLC filter processing.

When a message has passed through the DLC filter processing with the DRE bit in the RSCAN1GCFG register set to 0 (DLC replacement is disabled), the DLC value in the received message is stored in the buffer. In this case, all the data bytes in the received message are stored in the buffer.

When a message has passed through the DLC filter processing with the DRE bit in the RSCAN1GCFG register set to 1 (DLC replacement is enabled), the DLC value in the receive rule is stored in the buffer instead of the DLC value in the received message. In this case, a value of $00_H$ is stored in each data byte beyond the number of bytes which is indicated by the DLC value in the receive rule.

When the DLC value in the received message is smaller than that in the receive rule, the message does not pass through the DLC filter processing. In this case, the message is not stored in the receive buffer or the FIFO buffer and the DEF flag in the RSCAN1GERFL register is set to 1 (a DLC error is present).

#### 21.19.1.3 Routing Processing

Messages that passed through the acceptance filter processing and the DLC filter processing are stored in receive buffers, receive FIFO buffers, or transmit/receive FIFO buffers (set to receive mode). Message storage destination is set by the GAFLRMV and GAFLRMDP[6:0] bits in the RSCAN1GAFLP0j register (j = 0 to 15) and by the RSCAN1GAFLP1j register. Messages that passed through the acceptance filter processing and the DLC filter processing can be stored in up to eight buffers.

#### 21.19.1.4 Label Addition Processing

It is possible to add 12-bit label information to messages that passed through the filter processing and store them in buffers. This label information is set in the GAFLPTR[11:0] bits in the RSCAN1GAFLP0j register.

#### 21.19.1.5 Mirror Function Processing

The mirror function allows the CAN node to receive its own transmitted messages. The mirror function is made available by setting the MME bit in the RSCAN1GCFG register to 1 (mirror function is enabled).

When the mirror function is in use, receive rules for which the GAFLLB bit in the RSCAN1GAFLIDj register is set to 0 are used for data processing when receiving messages transmitted from other CAN nodes. When the CAN node is receiving its own transmitted messages, receive rules for which the GAFLLB bit is set to 1 are used for data processing.

### 21.19.1.6 Timestamp

The timestamp counter is a 16-bit free-running counter used for recording message receive time. The timestamp counter value is fetched at the start-of-frame (SOF) timing of a message and is then stored in a receive buffer or a FIFO buffer together with the message ID and data. Either pclk/2 or the CANm bit time clock (m = 6) may be selected as a timestamp counter clock source using the TSBTCS[2:0] and TSSS bits in the RSCAN1GCFG register. The timestamp counter count source is obtained by dividing the selected clock source by the TSP[3:0] value in the RSCAN1GCFG register.

When the CANm bit time clock is used as a clock source, the timestamp counter stops when the corresponding channel transitions to channel reset mode or channel halt mode. When the pclk/2 is used as a clock source, the timestamp function is not affected by channel mode.

The timestamp counter value is reset to $0000_H$ by setting the TSRST bit in the RSCAN1GCTR register to 1.

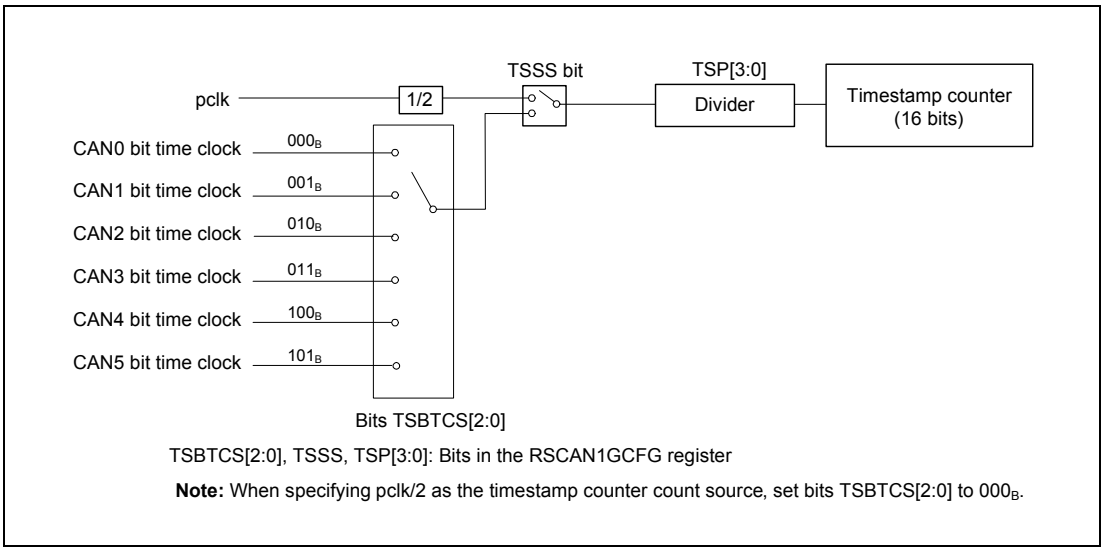



**Figure 21.46 Timestamp Function Block Diagram**

## 21.20 Transmission Functions

There are three types of transmission.

- Transmission using transmit buffers:
  Each channel has 16 buffers.
- Transmission using transmit/receive FIFO buffers (transmit mode):
  Each channel has three FIFO buffers. Up to 128 messages can be contained in a single FIFO buffer. Each FIFO buffer is used with a link to a transmit buffer. Only the message to be transmitted next in a FIFO buffer becomes the target of transmit priority determination. Messages are transmitted sequentially on a first-in, first-out basis.
- Transmission using transmit queues:
  Up to 16 transmit buffers per channel can be allocated to the transmit queues. Transmit buffer ((16 × i) + 15) is used as an access window of a corresponding channel. Transmit buffers are allocated to transmit queues in descending order of buffer number. All messages in transmit queues, which are targets of priority determination, are transmitted in the order of ID number.

**Figure 21.47** shows the allocation of transmit queues and transmit/receive FIFO buffer link.

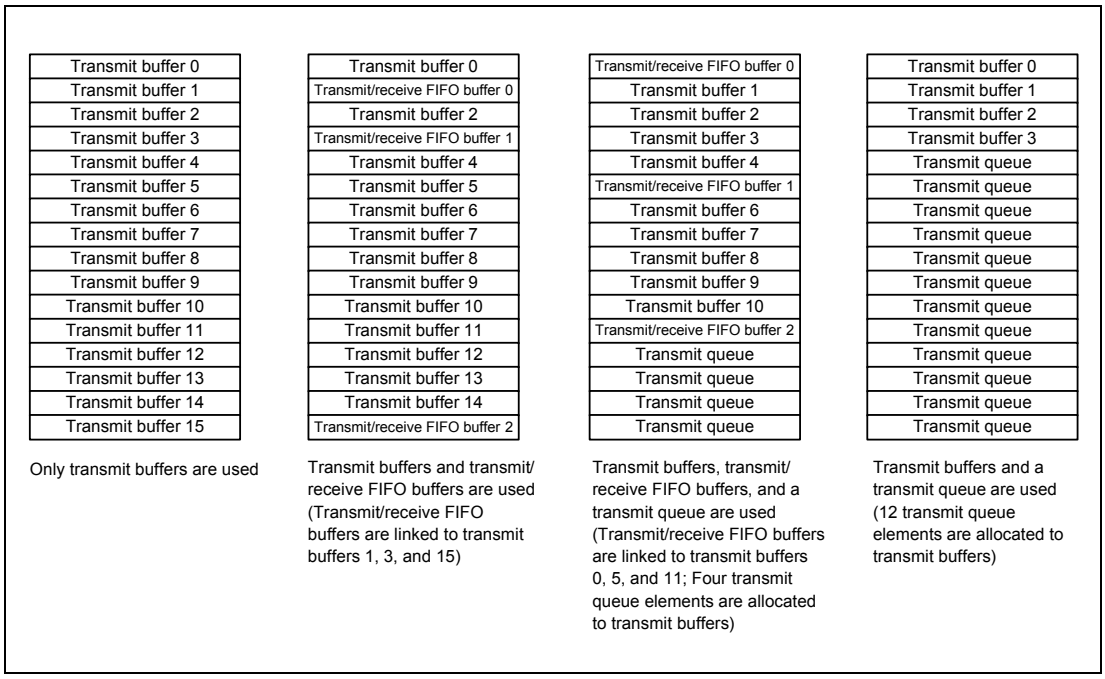



**Figure 21.47 Allocation of Transmit Queues and Transmit/Receive FIFO Buffer Links**

### 21.20.1 Transmit Priority Determination

If transmit requests are issued from multiple buffers or from the queue on the same channel, transmit priority is determined using one of the following methods.

The priority is determined by using one of the following methods.

- ID priority (TPRI bit = 0)
- Transmit buffer number priority (TPRI bit = 1)

All CAN channels use the setting of the TPRI bit in the RSCAN1GCFG register.

When the TPRI bit is set to 0, messages are transmitted according to the priority of stored message IDs. ID priority conforms to the CAN bus arbitration specification defined in the CAN specifications. All IDs of pending transmit messages are targets of priority determination, regardless of whether they are stored in transmit buffers, transmit/receive FIFO buffers (set to transmit mode), or the transmit queue. If even a single transmit queue is used, select ID priority. When transmit/receive FIFO buffers are used, the oldest message in a FIFO buffer becomes the target of priority determination. When a message is being transmitted from a transmit/receive FIFO buffer, the next message in the FIFO buffer becomes the target of priority determination. When a transmit queue is used, all messages in the transmit queue are targets of priority determination. If the same ID is set for two or more buffers, the buffer with the smaller buffer number takes precedence.

When the TPRI bit is set to 1, the message in the transmit buffer with the minimum buffer number among all buffers with a transmit request is transmitted first. When transmit/receive FIFO buffers are linked to transmit buffers, transmit priority is determined according to linked transmit buffer numbers.

When messages are retransmitted due to an arbitration-lost or an error, transmit priority determination is made again according to the TPRI bit.

### 21.20.2 Transmission Using Transmit Buffers

Setting the transmit request bit (TMTR bit in the RSCAN1TMCp register) in a transmit buffer to 1 (transmission is requested) allows transmission of data frames or remote frames.

The transmit result is shown by the TMTRF[1:0] flag in the corresponding RSCAN1TMSTSp register (p = 0 to i × 16 + 15). When transmit completes successfully, the TMTRF[1:0] flag is set to $10_B$ (transmission has been completed (without transmit abort request)) or $11_B$ (transmission has been completed (with transmit abort request)).

#### 21.20.2.1 Transmit Abort Function

With respect to transmit buffers for which the TMTRM bit in the RSCAN1TMSTSp register is set to 1 (a transmit request is present), when the TMTAR bit in the RSCAN1TMCp register is set to 1 (transmit abort is requested), the transmit request is canceled. When transmit abort is completed, the TMTRF[1:0] flag in the RSCAN1TMSTSp register is set to $01_B$ (transmit abort has been completed) and the transmit request is canceled (clearing the TMTRM bit to 0).

A message that is being transmitted or a message to be transmitted next according to the transmit priority determination cannot be aborted. However, when an arbitration-lost or an error occurs during transmission of a message for which the TMTAR bit is set to 1, retransmission is not performed.

#### 21.20.2.2 One-Shot Transmission Function (Retransmission Disabling Function)

When the TMOM bit in the RSCAN1TMCp register is set to 1 (one-shot transmission is enabled), transmission is performed only once. Even if an arbitration-lost or an error occurs, retransmission is not performed.

The one-shot transmit result is shown by the TMTRF[1:0] flag in the corresponding RSCAN1TMSTSp register. When one-shot transmission completes successfully, the TMTRF[1:0] flag is set to $10_B$ or $11_B$. When an arbitration-lost or an error occurs, the TMTRF[1:0] flag is set to $01_B$ (transmit abort has been completed).

### 21.20.3 Transmission Using FIFO Buffers

Multiple messages can be stored in a single transmit/receive FIFO buffers, up to the number specified by the FIFO buffer depth, which is set by the CFDC[2:0] bits in the RSCAN1CFCCk register (k = 0 to i × 3 + 2). Messages are transmitted sequentially on a first-in, first-out basis.

Each transmit/receive FIFO buffer is linked to a transmit buffer selected by the CFTML[3:0] bits in the RSCAN1CFCCk register. When the CFE bit in the RSCAN1CFCCk register is set to 1 (transmit/receive FIFO buffers are used), transmit/receive FIFO buffers become targets of transmit priority determination. Priority of only the next transmit message is determined in the FIFO buffer.

When the CFE bit is set to 0 (no transmit/receive FIFO buffer is used), the CFEMP flag is set to 1 (the transmit/receive FIFO buffer contains no message (buffer empty)) at the timing below.

- The transmit/receive FIFO buffer becomes empty immediately if the message in it is not being transmitted or is not to be transmitted next.
- The transmit/receive FIFO buffer becomes empty after transmission completion, CAN bus error detection, or arbitration-lost in the case that a message in it is being transmitted or to be transmitted next.

When the CFE bit is cleared to 0, all messages in transmit/receive FIFO buffers are lost and messages cannot be stored in FIFO buffers. Confirm that the CFEMP flag is set to 1 before setting the CFE bit to 1 again.

### 21.20.3.1 Interval Transmission Function

A message transmission interval time can be set to space the transmission of messages from the same FIFO buffer when using a transmit/receive FIFO buffer set to transmit mode.

Immediately after the first message has been transmitted successfully from the FIFO buffer with the CFE bit in the RSCAN1CFCCk register set to 1, the interval timer starts counting (after EOF7 of the CAN protocol). After that, when the interval time has passed, the next message is transmitted. The interval timer stops in channel reset mode or by clearing the CFE bit to 0.

The interval time is set by the CFITT[7:0] bits in the RSCAN1CFCCk register. When the interval timer is not used, set the CFITT[7:0] bits to $00_H$.

Select an interval timer count source using the CFITR and CFITSS bits in the RSCAN1CFCCk register. When the CFITR and CFITSS bits are set to $00_B$, the count source is obtained by dividing pclk/2 by the value of the ITRCP[15:0] bits in the RSCAN1GCFG register. When the CFITR and CFITSS bits are set to $10_B$, the count source is obtained by dividing pclk/2 by (the value of the ITRCP[15:0] bits × 10). When the CFITR and CFITSS bits are set to $x1_B$, the CANm bit time clock is used as a count source.

The interval time is calculated by the following equations where M is the value of ITRCP[15:0] and N is the value of CFITT[7:0].

- When CFITR and CFITSS = $00_B$ (fPBA is the frequency of pclk):

$$\frac{1}{f_{PBA}} \times 2 \times M \times N$$

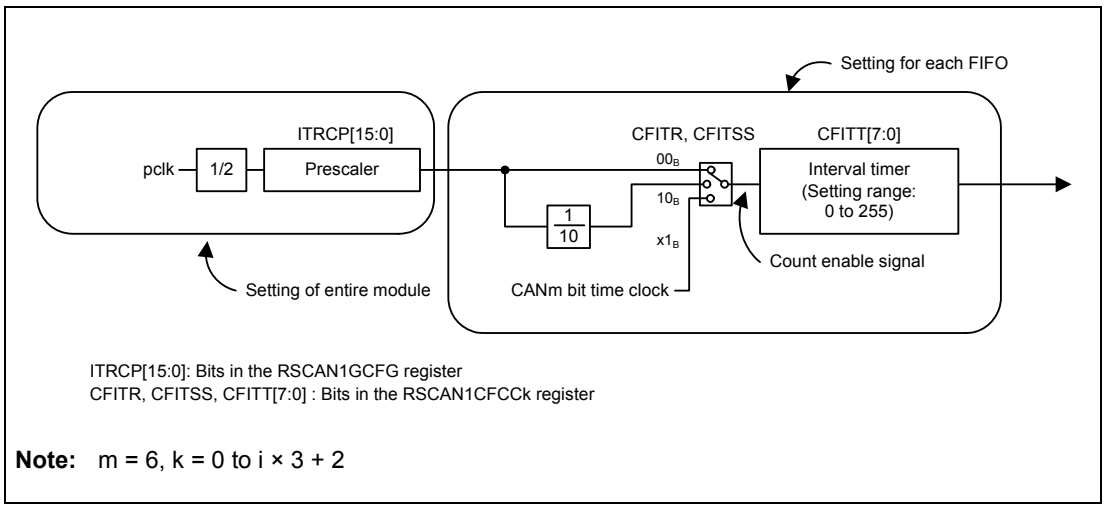
- When CFITR and CFITSS = $10_B$:

$$\frac{1}{f_{PBA}} \times 2 \times M \times 10 \times N$$

- When CFITR and CFITSS = $x1_B$ (fCANBIT is the frequency of CANm bit time clock):

$$\frac{1}{f_{CANBIT}} \times N$$

**Figure 21.48** shows the interval timer block diagram.




**Note:** m = 6, k = 0 to i × 3 + 2

**Figure 21.48 Interval Timer Block Diagram**

**Figure 21.49** shows the interval timer timing diagram.

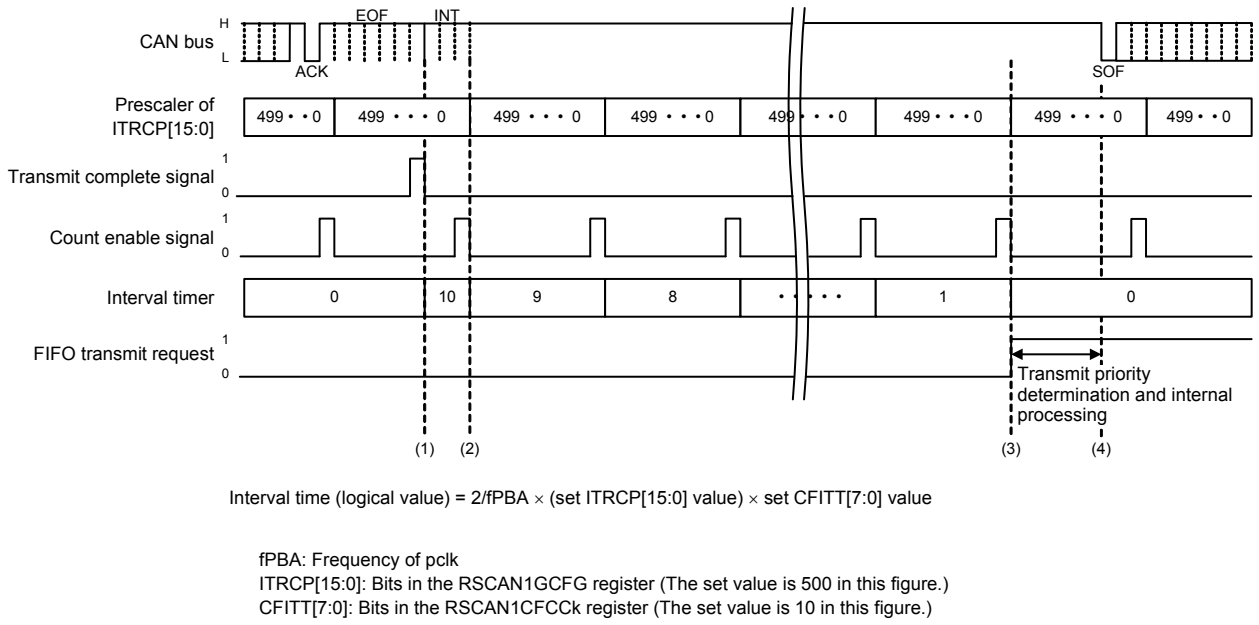



**Figure 21.49 Interval Timer Timing Chart**

(1) The interval timer starts counting upon completion of transmission. Since the prescaler is not initialized at the time of transmission completion, the first interval time contains an error of up to one count of the interval timer.

(2) The interval timer is decremented by the next count enable signal.

(3) When the interval timer has decreased to 0, the transmit/receive FIFO buffer issues a transmit request.

(4) The transmit/receive FIFO buffer is determined for the next transmission by the priority determination, it starts transmitting data. Transmission starts usually with a delay of three CANm bit time clock cycles or less from the issue of transmit request. If multiple internal processes (such as receive filter processing, message routing, and transmit priority determination) take place in all channels, a delay of up to 348 cycles of the pclk may be generated.

### 21.20.4 Transmission Using Transmit Queues

Three to sixteen buffers are allocated to a transmit queue for each channel, and transmit buffer ((16 × i) + 15) is used as an access window of a corresponding channel.

All messages in a transmit queue are targets of transmit priority determination and are transmitted in the ID priority order regardless of storage sequence. If two messages having the same ID are stored in a transmit queue, these messages are not always transmitted in the order of their storage in the transmit queue.

Setting the TXQE bit in the RSCAN1TXQCCm register to 0 disables transmit queues. When the TXQE bit is set to 0, the TXQEMP flag in the RSCAN1TXQSTSm register is set to 1 (the transmit queue contains no messages (transmit queue empty)) at the timing below.

- The transmit queue becomes empty immediately when no message in it is being transmitted or will be transmitted next.
- The transmit queue becomes empty after transmission completion, CAN bus error detection, or arbitration-lost when a message in it is being transmitted or will be transmitted next.

When the TXQE bit is cleared to 0, all messages in transmit queues are lost and messages cannot be stored in transmit queues. Confirm that the TXQEMP flag is set to 1 before setting the TXQE bit to 1 again.

### 21.20.5 Transmit History Function

Information about transmission-completed messages can be stored in the transmit history buffer. Each channel has a single transmit history buffer that can contain 16 sets of transmit history data.

A message transmit source buffer type can be selected by the THLDTE bit in the RSCAN1THLCCm register. The THLEN bit in the RSCAN1CFIDk register (k = 0 to i × 3 + 2) determines whether transmit history data is stored for each message.

The following information on a transmitted message will be stored in the transmission history buffer after the successful completion of transmission.

Storage of the transmission history data after the successful completion of transmission may take up to 123 cycles of pclk.

- Buffer type — $001_B$: Transmit buffer<br>$010_B$: Transmit/receive FIFO buffer<br>$100_B$: Transmit queue
- Buffer number — Number of source transmit buffer, transmit queue, or transmit/receive FIFO buffer. This number depends on buffer types. See **Table 21.294**.
- Label data — Label information of the transmit message

Table 21.294 Transmit History Data Buffer Numbers

| Buffer No. / Buffer type | $001_B$ | $010_B$ | $100_B$ |
|---|---|---|---|
| $0000_B$ | Transmit buffer 16 × i + 0 | Buffer numbers of the transmit buffer linked to the transmit/receive FIFO buffer by the CFTML[3:0] bits in the RSCAN1CFCCk register (k = 0 to i × 3 + 2) | Buffer numbers of the transmit buffer allocated to the transmit queue that performed transmission |
| $0001_B$ | Transmit buffer 16 × i + 1 | | |
| $0010_B$ | Transmit buffer 16 × i + 2 | | |
| $0011_B$ | Transmit buffer 16 × i + 3 | | |
| $0100_B$ | Transmit buffer 16 × i + 4 | | |
| $0101_B$ | Transmit buffer 16 × i + 5 | | |
| $0110_B$ | Transmit buffer 16 × i + 6 | | |
| $0111_B$ | Transmit buffer 16 × i + 7 | | |
| $1000_B$ | Transmit buffer 16 × i + 8 | | |
| $1001_B$ | Transmit buffer 16 × i + 9 | | |
| $1010_B$ | Transmit buffer 16 × i + 10 | | |
| $1011_B$ | Transmit buffer 16 × i + 11 | | |
| $1100_B$ | Transmit buffer 16 × i + 12 | | |
| $1101_B$ | Transmit buffer 16 × i + 13 | | |
| $1110_B$ | Transmit buffer 16 × i + 14 | | |
| $1111_B$ | Transmit buffer 16 × i + 15 | | |

Label data is used to identify each message. Unique label data can be added to each message transmitted from a transmit buffer, transmit queue, or transmit/receive FIFO buffer.

Transmit history data can be read from the RSCAN1THLACCm register. If an attempt is made to store new transmit history data while the buffer is full, the buffer overflows and the new data is discarded.

## 21.21 Test Function

The test function is classified into communication tests and global tests.

- Communication tests: Performed for each channel.
  - Standard test mode
  - Listen-only mode
  - Self-test mode 0 (external loopback mode)
  - Self-test mode 1 (internal loopback mode)
- Global tests: Performed for the entire module
  - RAM test (read/write test)

### 21.21.1 Standard Test Mode

Standard test mode allows CRC test.

### 21.21.2 Listen-Only Mode

Listen-only mode allows reception of data frames and remote frames. Only recessive bits are transmitted on the CAN bus, and the ACK bit, overload flag, and active error flag are not transmitted.

Listen-only mode is available for detecting the communication speed.

Do not make a transmit request from any buffer or queue in listen-only mode.

**Figure 21.50** shows the connection when listen-only mode is selected.

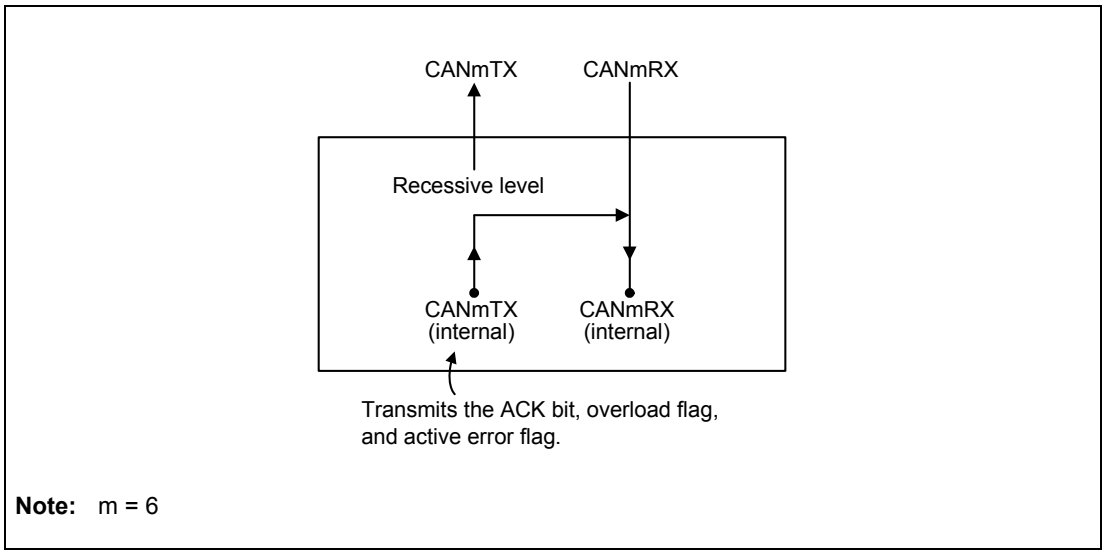



**Figure 21.50 Connection when Listen-Only Mode is Selected**

### 21.21.3 Self-Test Mode (Loopback Mode)

In self-test mode, transmitted messages are compared with the receive rule of the own channel and the messages are stored in a buffer if they have passed through the filter processing. Messages transmitted from other CAN nodes are compared only with the receive rule for which the GAFLLB bit in the RSCAN1GAFLIDj register (j = 0 to 15) is set to 0 (when a message transmitted from another CAN node is received).

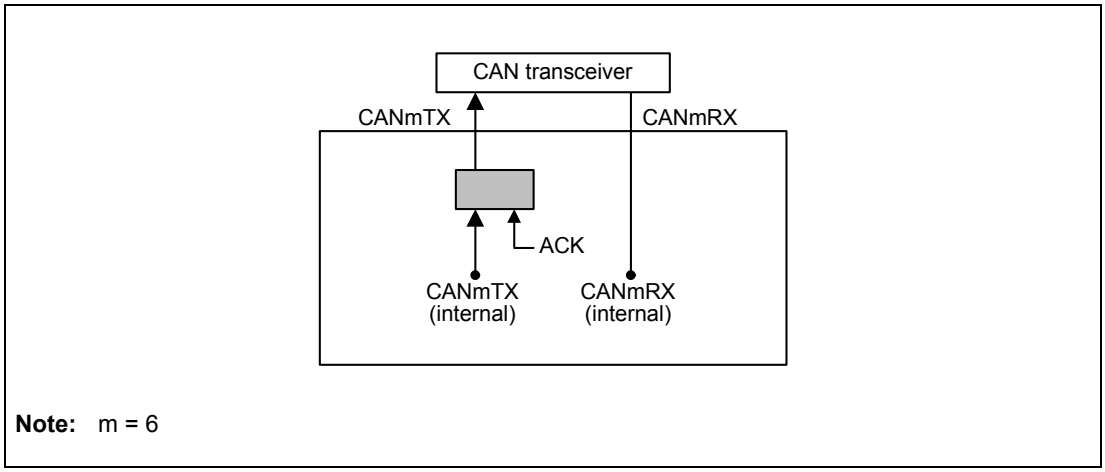
If the mirror function and self-test mode are both enabled, the self-test mode setting takes precedence.

#### 21.21.3.1 Self-Test Mode 0 (External Loopback Mode)

Self-test mode 0 is used to perform a loopback test within a channel including the CAN transceiver.

In self-test mode 0, transmitted messages are handled as messages received through the CAN transceiver and are stored in a buffer. An ACK bit is generated to receive messages transmitted from the own CAN node.

**Figure 21.51** shows the connection when self-test mode 0 is selected.




**Note:** m = 6

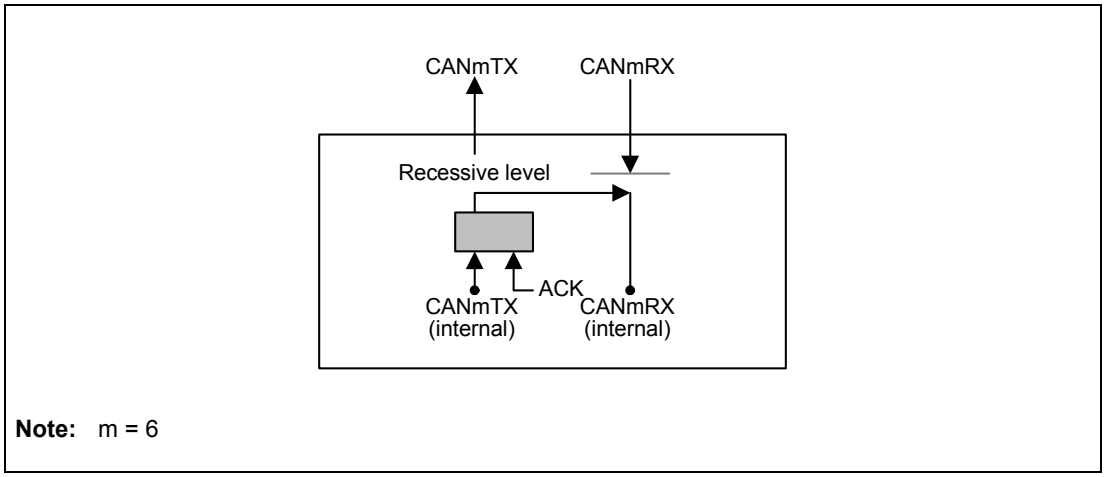
**Figure 21.51 Connection when Self-Test Mode 0 is Selected**

### 21.21.3.2 Self-Test Mode 1 (Internal Loopback Mode)

In self-test mode 1, transmitted messages are handled as received messages and are stored in a buffer. An ACK bit is generated to receive messages transmitted from the own CAN node.

In self-test mode 1, internal feedback from the internal CANmTX pin (m = 6) to the internal CANmRX pin is performed. The external CANmRX pin input is isolated. The external CANmTX pin outputs only recessive bits.

**Figure 21.52** shows the connection when self-test mode 1 is selected.




**Note:** m = 6

**Figure 21.52 Connection when Self-Test Mode 1 is Selected**

## 21.21.4 RAM Test

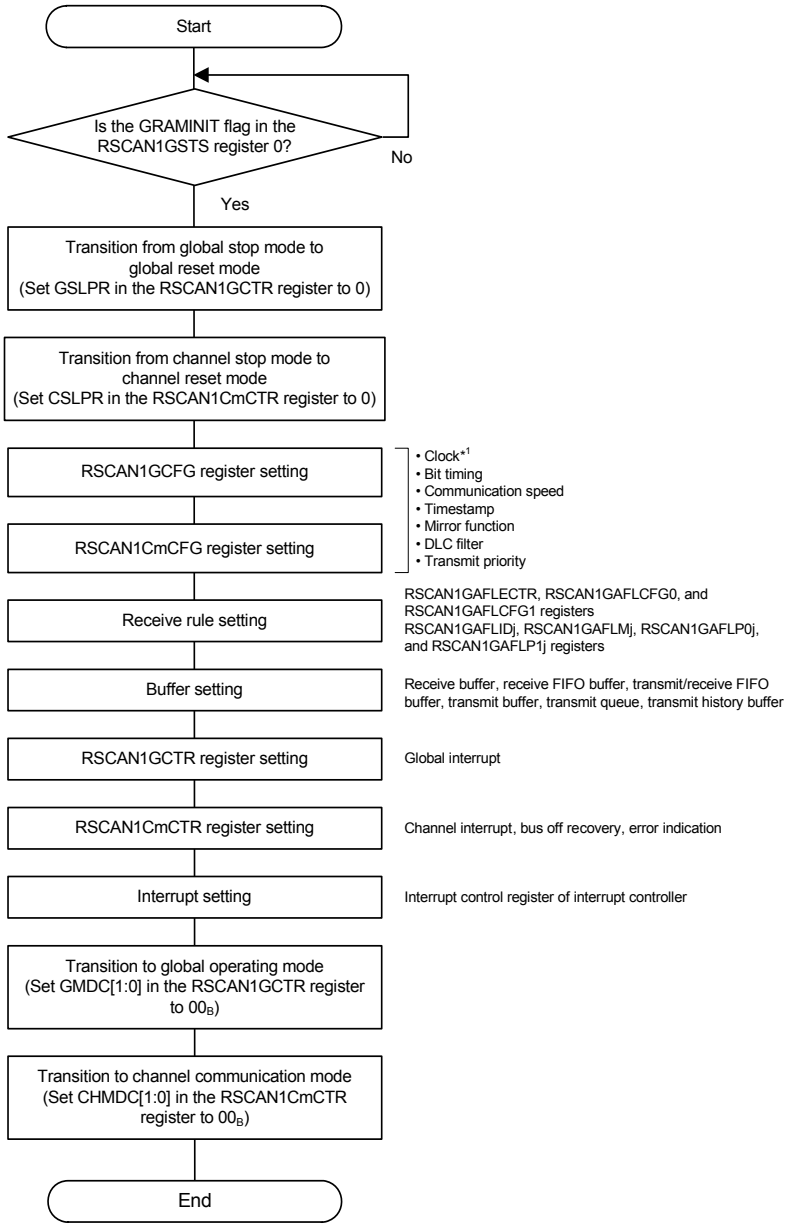
The RAM test function allows accesses to all CAN RAM addresses.

When the RAM test function is used, the RAM is divided into pages of 256 bytes each. RAM test page is set by the RTMPS[6:0] bits in the RSCAN1GTSTCFG register. Data in the set page can be read from and written to the RSCAN1RPGACCr register (r = 0 to 63). The available total RAM size is 7296 bytes ($1C80_H$).

# 21.22 RS-CAN Setting Procedure

## 21.22.1 Initial Settings

The RS-CAN module initializes the CAN RAM after the MCU is reset. The RAM initialization time is 3650 cycles of the pclk. The GRAMINIT flag in the RSCAN1GSTS register is set to 1 (CAN RAM initialization is ongoing) during the RAM initialization and is cleared to 0 (CAN RAM initialization is finished) when the initialization is completed. Make CAN settings after the GRAMINIT flag is cleared to 0. **Figure 21.53** shows the CAN setting procedure after the MCU is reset.




Note 1. For the setting of CAN clock frequency, see the **Table 21.208, Range of Operating Frequency Depending on the Transfer Rate and the Number of Channels in Use in the RH850/F1K**.

**Note:** m = 6, j = 0 to 15

**Figure 21.53 CAN Setting Procedure after the MCU is Reset**

### 21.22.1.1 Clock Setting

Set the CAN clock (fCAN) as a clock source of the RS-CAN module. Select the clk_xincan or clkc using the DCS bit in the RSCAN1GCFG register.

### 21.22.1.2 Bit Timing Setting

In the CAN protocol, one bit of a communication frame consists of three segments SS, TSEG1, and TSEG2, of which two segments TSEG1 and TSEG2 can be set by the RSCAN1CmCFG register for each channel. Sample point timing can be determined by setting these two segments. This timing can be adjusted in units of 1 Time Quantum (hereafter referred to as Tq). 1 Tq is equal to one CANmTq clock cycle. The CANmTq clock is obtained by selecting the clock source with the DCS bit in the RSCAN1GCFG register and selecting the clock division ratio with the BRP[9:0] bits in the RSCAN1CmCFG register.

**Figure 21.54** shows the bit timing chart. **Table 21.295** shows an example of bit timing setting.

SS = 1 Tq fixed
Set TSEG1 to a range of 4 Tq to 16 Tq
Set TSEG2 to a range of 2 Tq to 8 Tq
Set SJW to a range of 1 Tq to 4 Tq
Set SS + TSEG1 + TSEG2 to a range of 8 Tq to 25 Tq
TSEG1 > TSEG2 ≥ SJW

- SS (synchronization segment):
  The SS is a segment that performs synchronization by monitoring the edge from recessive to dominant bits in the Interframe Space.
  Interframe Space consists of Intermission, Suspend Transmission, and Bus Idle. All nodes can start transmission during Bus Idle.
- TSEG1 (Time segment 1):
  TSEG1 is a segment that absorbs physical delay on the CAN network. The physical delay on the CAN network is twice the total of the delay on the CAN bus, the delay in the input comparator, and the delay in the output driver.
- TSEG2 (Time segment 2):
  TSEG2 is a segment that compensates phase error due to an error in frequency.
- SJW (Resynchronization jump width)
  The SJW is a length to extend or reduce the time segment to compensate for an error in phase due to phase error.

**Figure 21.54 Bit Timing Chart**

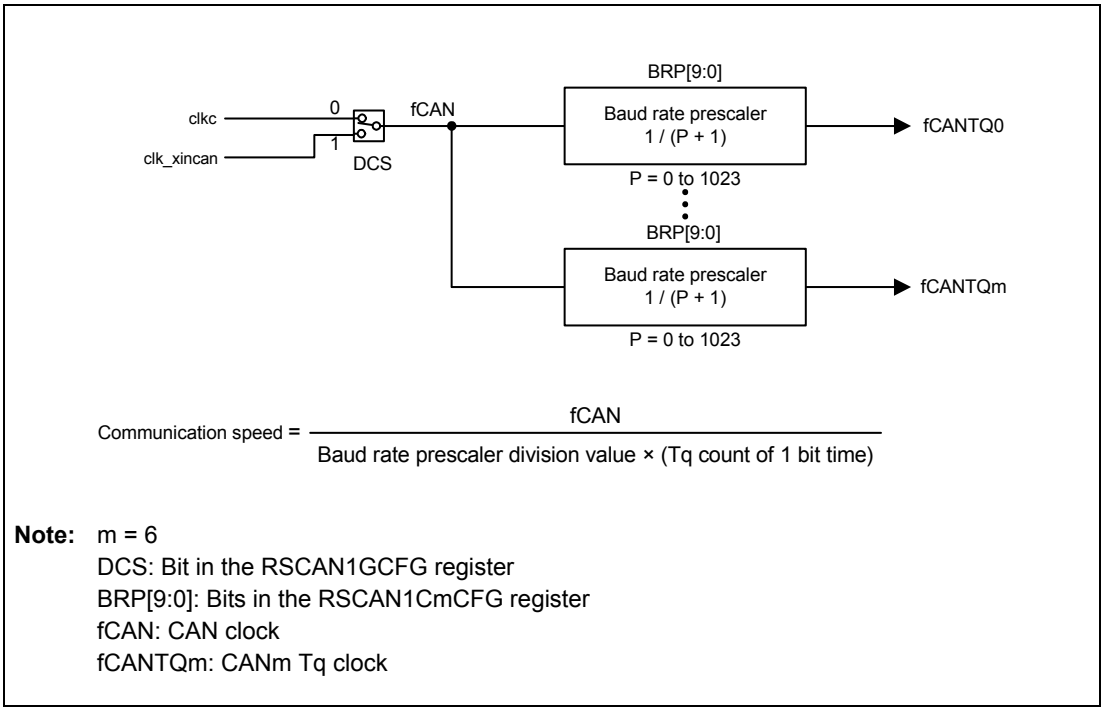
**Table 21.295 Example of Bit Timing Setting**

| 1 Bit | Set Value (Tq) | | | | Sample Point (%) Note: See Figure 21.54. |
|---|---|---|---|---|---|
| | SS | TSEG1 | TSEG2 | SJW | |
| 8 Tq | 1 | 4 | 3 | 1 | 62.50 |
| | 1 | 5 | 2 | 1 | 75.00 |
| 10 Tq | 1 | 6 | 3 | 1 | 70.00 |
| | 1 | 7 | 2 | 1 | 80.00 |
| 16 Tq | 1 | 10 | 5 | 1 | 68.75 |
| | 1 | 11 | 4 | 1 | 75.00 |
| 20 Tq | 1 | 12 | 7 | 1 | 65.00 |
| | 1 | 13 | 6 | 1 | 70.00 |

### 21.22.1.3 Communication Speed Setting

Set the CAN communication speed for each channel using the fCAN, baud rate prescaler division value (BRP[9:0] bits in the RSCAN1CmCFG register), and Tq count per bit time.

**Figure 21.55** shows the CAN clock control block diagram, and **Table 21.296** shows an example of the communication speed setting.




**Note:** m = 6
DCS: Bit in the RSCAN1GCFG register
BRP[9:0]: Bits in the RSCAN1CmCFG register
fCAN: CAN clock
fCANTQm: CANm Tq clock

**Figure 21.55 CAN Clock Control Block Diagram**

**Table 21.296 Example of Communication Speed Setting**

| fCAN / Communication speed | 40 MHz | 32 MHz | 24 MHz | 16 MHz | 8 MHz |
|---|---|---|---|---|---|
| 1 Mbps | 8 Tq (5)<br>20 Tq (2) | 8 Tq (4)<br>16 Tq (2) | 8 Tq (3)<br>12 Tq (2)<br>24 Tq (1) | 8 Tq (2)<br>16 Tq (1) | 8 Tq (1) |
| 500 Kbps | 8 Tq (10)<br>20 Tq (4) | 8 Tq (8)<br>16 Tq (4) | 8 Tq (6)<br>12 Tq (4)<br>24 Tq (2) | 8 Tq (4)<br>16 Tq (2) | 8 Tq (2)<br>16 Tq (1) |
| 250 Kbps | 8 Tq (20)<br>20 Tq (8) | 8 Tq (16)<br>16 Tq (8) | 8 Tq (12)<br>12 Tq (8)<br>24 Tq (4) | 8 Tq (8)<br>16 Tq (4) | 8 Tq (4)<br>16 Tq (2) |
| 125 Kbps | 8 Tq (40)<br>20 Tq (16) | 8 Tq (32)<br>16 Tq (16) | 8 Tq (24)<br>12 Tq (16)<br>24 Tq (8) | 8 Tq (16)<br>16 Tq (8) | 8 Tq (8)<br>16 Tq (4) |

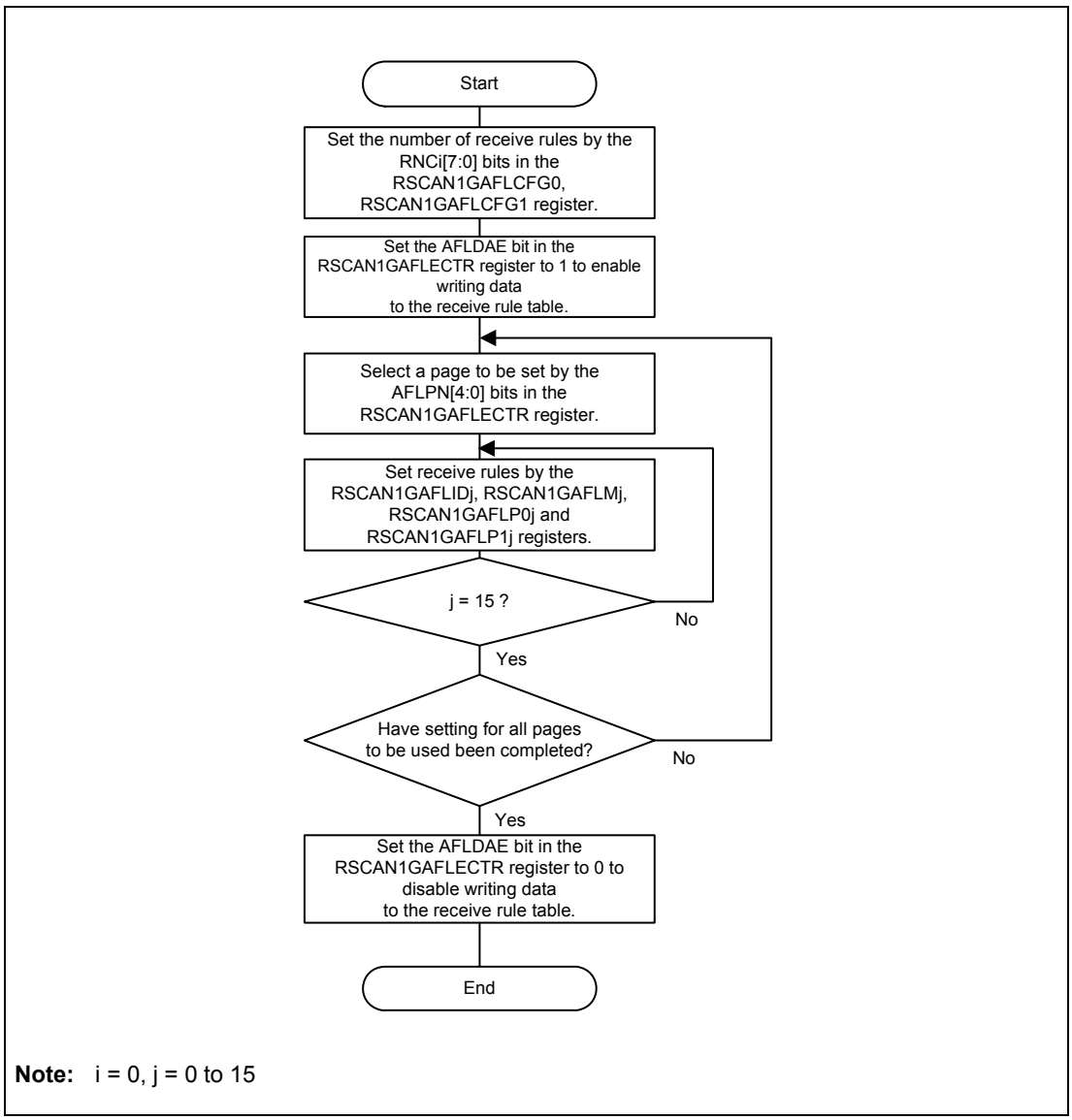
**Note:** Values in ( ) are baud rate prescaler division values.

### 21.22.1.4 Receive Rule Setting

Receive rules can be set using receive rule-related registers.

Up to 16 receive rules can be registered per page. Specify pages 0 to 23 by the AFLPN[4:0] bits in the RSCAN1GAFLECTR register. Set receive rule table write enable/disable using the AFLDAE bit.

**Figure 21.56** shows the receive rule setting procedure.



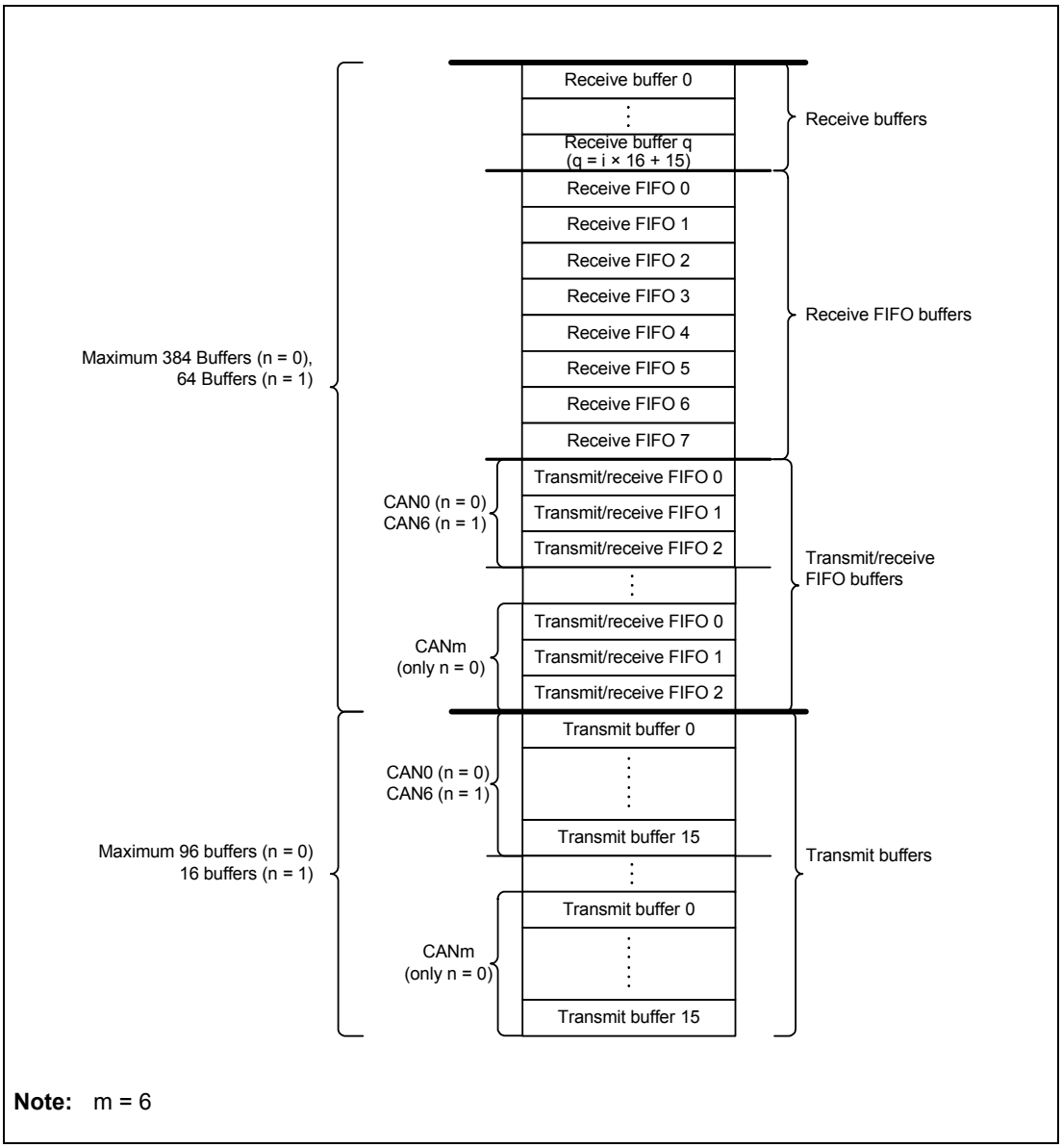

**Note:** i = 0, j = 0 to 15

**Figure 21.56 Receive Rule Setting Procedure**

### 21.22.1.5 Buffer Setting

Set sizes and interrupt sources of buffers. For transmit/receive FIFO buffers that are set to transmit mode, set transmit buffers to be linked.

**Figure 21.57** shows the buffer configuration. **Figure 21.58** shows the buffer setting procedure.




**Note:** m = 6

**Figure 21.57 Buffer Configuration**

**CAUTION**

Receive buffers, receive FIFO buffers, transmit/receive FIFO buffers, and transmit buffers are located in succession.

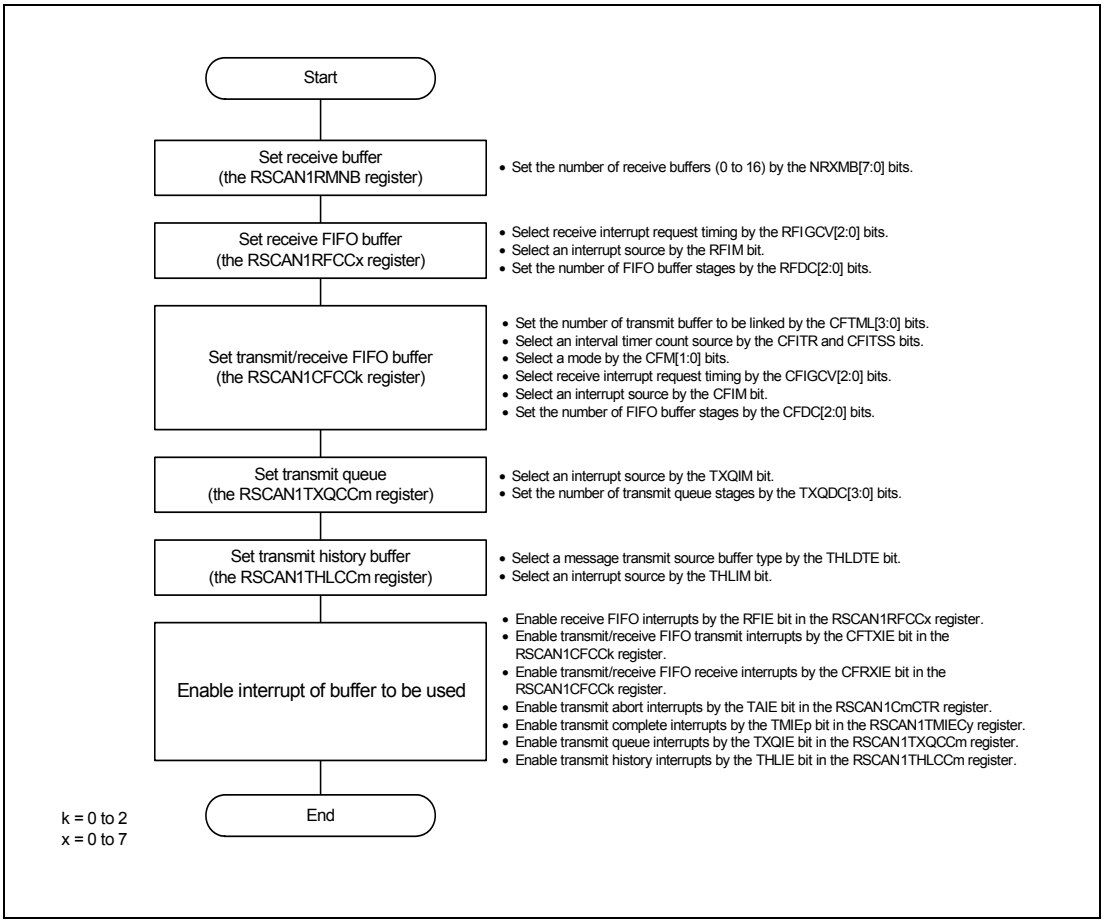
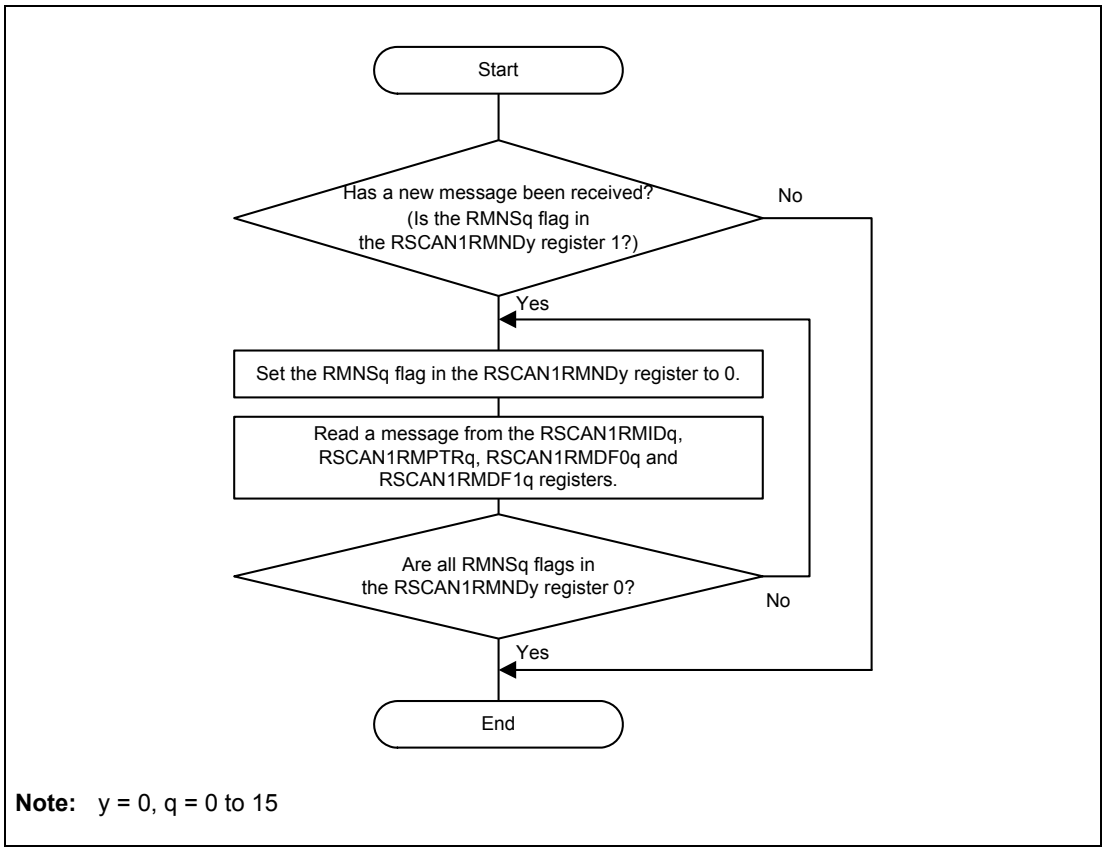



**Figure 21.58 Buffer Setting Procedure**

## 21.22.2 Reception Procedure

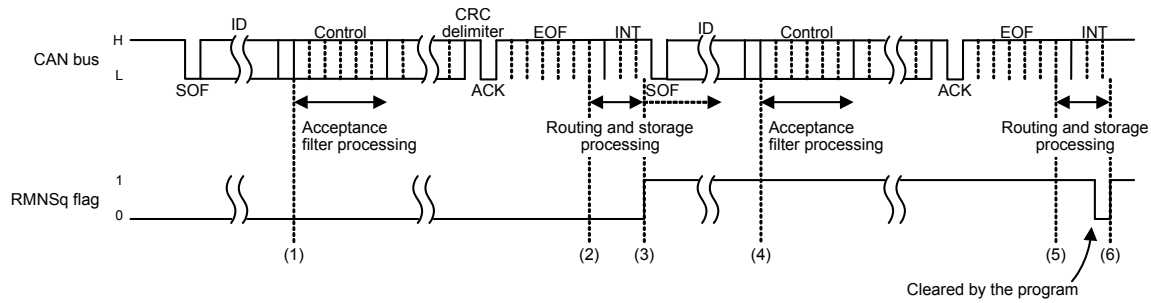
### 21.22.2.1 Receive Buffer Reading Procedure

When the processing to store received messages in a receive buffer starts, the RMNSq flag in the RSCAN1RMNDy register ((y = 0, q = 0 to 15) is set to 1 (receive buffer q contains a new message). Messages can be read from the RSCAN1RMIDq, RSCAN1RMPTRq, RSCAN1RMDF0q, and RSCAN1RMDF1q registers. If the next message has been received before the current message is read from the receive buffer, the message is overwritten. **Figure 21.59** shows the receive buffer reading procedure.




**Note:** y = 0, q = 0 to 15

**Figure 21.59 Receive Buffer Reading Procedure**




**Note:** y = 0, q = 0 to 15
RMNSq: Flag in the RSCAN1RMNDy register

**Figure 21.60 Receive Buffer Reception Timing Chart**

(1) When the ID field in a message has been received, the acceptance filter processing starts.

(2) When the message matches the receive rule of the corresponding channel and the message has been successfully received, the routing processing to transfer the message to the specified buffer starts. When the DCE bit in the RSCAN1GCFG register is set to 1 (DLC check is enabled), the DLC filter processing starts at this time.

(3) When the message has passed through the DLC filter processing, the processing to store the message in the specified receive buffer starts.
When the message storage processing starts, the RMNSq flag in the corresponding RSCAN1RMNDy register is set to 1 (the receive buffer contains a new message). If other channels are performing filter processing or transmit priority determination processing, the routing processing and the storage processing may be delayed.

(4) When the ID field of the next message has been received, the acceptance filter processing starts.

(5) When the message matches the receive rule of the corresponding channel and the message has been successfully received, the routing processing to transfer the message to the specified buffer starts. When the DCE bit in the RSCAN1GCFG register is set to 1 (DLC check is enabled), the DLC filter processing starts at this time.

(6) When the corresponding RMNSq flag is cleared to 0 (the receive buffer contains no new message), this flag is set to 1 again when the message storage processing starts. Even if the RMNSq flag remains 1, a new message is overwritten to the receive buffer. The RMNSq flag should not be cleared to 0 during storage of messages.

### 21.22.2.2 FIFO Buffer Reading Procedure

When received messages have been stored in one or more receive FIFO buffers or a transmit/receive FIFO buffer that is set to receive mode, the corresponding message count display counter (RFMC[7:0] bits in the RSCAN1RFSTSx register (x = 0 to 7) or CFMC[7:0] bits in the RSCAN1CFSTSk register (k = 0 to 2)) is incremented. At this time, when the RFIE bit (receive FIFO interrupt is enabled) in the RSCAN1RFCCx register or the CFRXIE bit (transmit/receive FIFO receive interrupt is enabled) in the RSCAN1CFCCk register is set to 1, an interrupt request is generated. Received messages can be read from the RSCAN1RFIDx, RSCAN1RFPTRx, RSCAN1RFDF0x, and RSCAN1RFDF1x registers for receive FIFO buffers, or from the RSCAN1CFIDk, RSCAN1CFPTRk, RSCAN1CFDF0k, and RSCAN1CFDF1k registers for transmit/receive FIFO buffers. Messages in FIFO buffers can be read sequentially on a first-in, first-out basis.

When the message count display counter value matches the FIFO buffer depth (a value set by the RFDC[2:0] bits in the RSCAN1RFCCx register or the CFDC[2:0] bits in the RSCAN1CFCCk register), the RFFLL or CFFLL flag is set to 1 (the receive FIFO buffer is full).

When all messages have been read out of the FIFO buffer, the RFEMP flag in the RSCAN1RFSTSx register or the CFEMP flag in the RSCAN1CFSTSk register is set to 1 (the receive FIFO buffer contains no unread message (buffer empty)).

If the RFE bit or the CFE bit is cleared to 0 (no receive FIFO buffer is used) with the interrupt request flag (RFIF flag in the RSCAN1RFSTSx register or CFRXIF flag in the RSCAN1CFSTSk register) set to 1 (a receive FIFO interrupt request is present), the interrupt request flag is not automatically cleared to 0. The program must clear the interrupt request flag to 0.

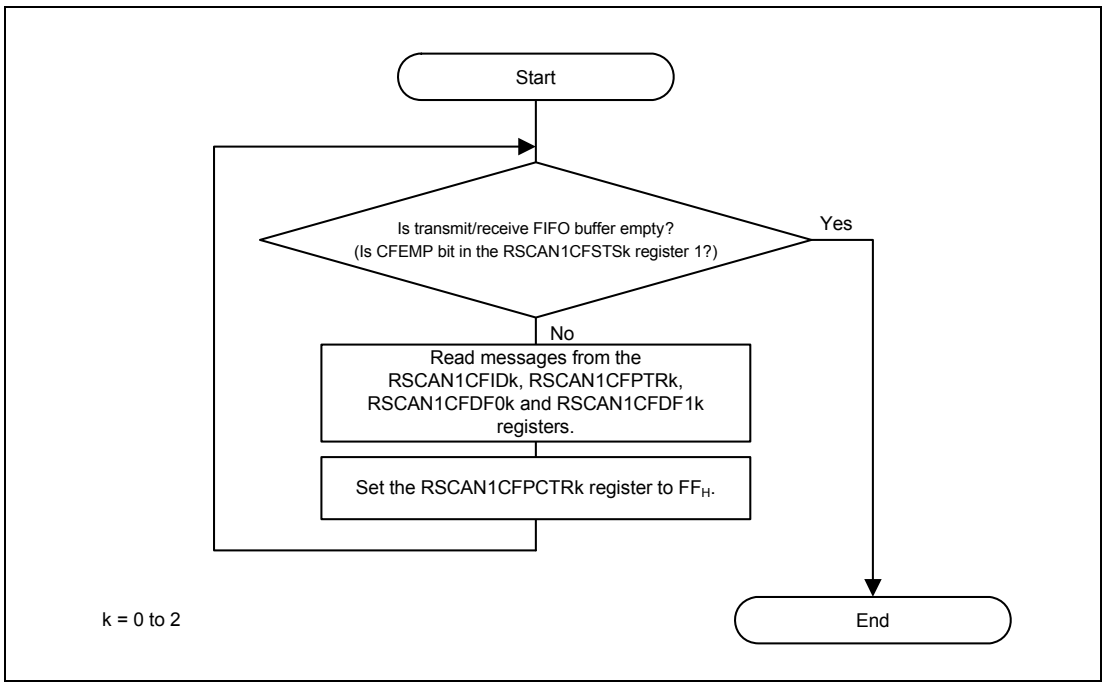



**Figure 21.61 Transmit/Receive FIFO Buffer Reading Procedure**

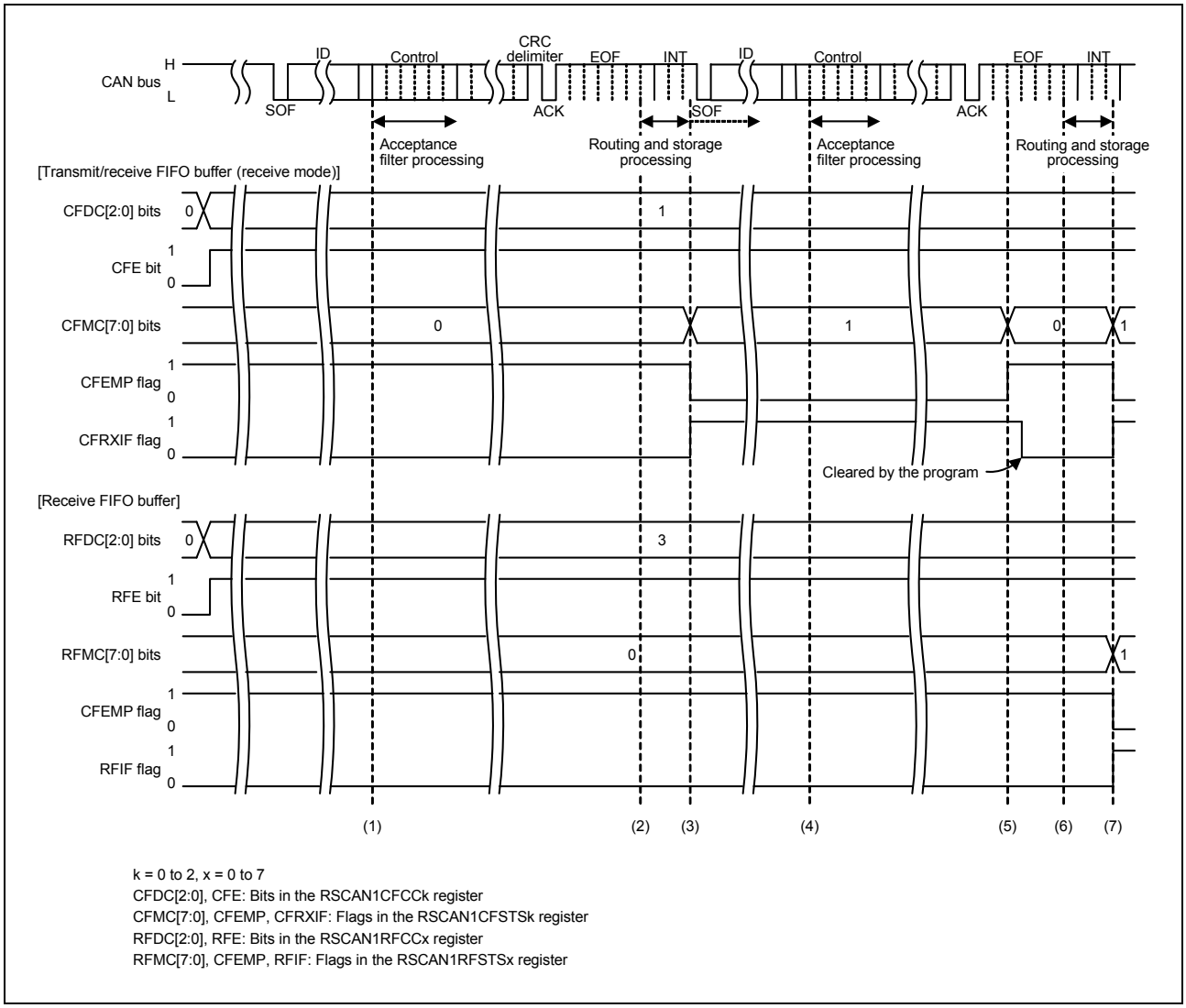



**Figure 21.62 FIFO Buffer Reception Timing Chart**

(1) When the ID field in a message has been received, the acceptance filter processing starts.

(2) When the message matches the receive rule of the corresponding channel and the message has been successfully received, the routing processing to transfer the message to the specified buffer starts. When the DCE bit in the RSCAN1GCFG register is set to 1 (DLC check is enabled), the DLC filter processing starts at this time.

(3) When the message has passed through the DLC filter processing and the CFE bit in the RSCAN1CFCCk register is 1 (transmit/receive FIFO buffers are used) and the CFDC[2:0] value in the RSCAN1CFCCk register is $001_B$ or more, the message is stored in the transmit/receive FIFO buffer that is set to receive mode. The CFMC[7:0] value in the RSCAN1CFSTSk register is incremented and becomes $01_H$. When the CFIM bit in the RSCAN1CFCCk register is set to 1 (a FIFO receive interrupt request is generated each time a message has been received), the CFRXIF flag in the RSCAN1CFSTSk register is set to 1 (a transmit/receive FIFO receive interrupt request is present). The CFRXIF flag can be reset to 0 by the program.

(4) When the ID field of the next message has been received, the acceptance filter processing starts.

(5) Read received messages from the RSCAN1CFIDk, RSCAN1CFPTRk, RSCAN1CFDF0k, and RSCAN1CFDF1k registers and write $FF_H$ to the RSCAN1CFPCTRk register. This causes the

CFMC[7:0] bits in the RSCAN1CFSTSk register to be decremented. When CFMC[7:0] becomes $00_H$, the CFEMP flag in the RSCAN1CFSTSk register becomes 1 (the transmit/receive FIFO buffer contains no message (buffer empty)).

(6) When the message matches the receive rule of the corresponding channel and the message has been successfully received, the routing processing to transfer the message to the specified buffer starts. When the DCE bit in the RSCAN1GCFG register is set to 1 (DLC check is enabled), the DLC filter processing starts at this time.

(7) The message is stored in the transmit/receive FIFO buffer set in receive mode when the message has passed through the DLC filter process if the CFE bit is set to 1 (transmit/receive FIFO buffers are used), and the CFDC[2:0] bits are set to $001_B$ or more. The CFMC[7:0] bit value is incremented by 1 to be $01_H$. When the CFIM bit is set to 1 (an interrupt occurs each time a message has been received), the CFRXIF flag is set to 1 (a transmit/receive FIFO receive interrupt request is present).
The message is stored in the receive FIFO buffer if the RFE bit in the RSCAN1RFCCx register is set to 1 (receive FIFO buffers are used), and the RFDC[2:0] bits in the RSCAN1RFCCx register are set to $001_B$ or more. The RFMC[7:0] bits in the RSCAN1RFSTSx register are set to $01_H$ by being incremented by 1. When the RFIM bit in the RSCAN1RFCCx register is set to 1 (an interrupt occurs each time a message has been received), the RFIF flag in the RSCAN1RFSTSx register is set to 1 (a receive FIFO interrupt request is present).

## 21.22.3 Transmission Procedure

### 21.22.3.1 Procedure for Transmission from Transmit Buffers

**Figure 21.63** shows the procedure for transmission from transmit buffers.

**Figure 21.64** shows a timing chart where messages are transmitted from two transmit buffers in the same channel and transmission has been successfully completed. **Figure 21.65** shows a timing chart where messages are transmitted from two transmit buffers in the same channel and transmit abort has been completed.

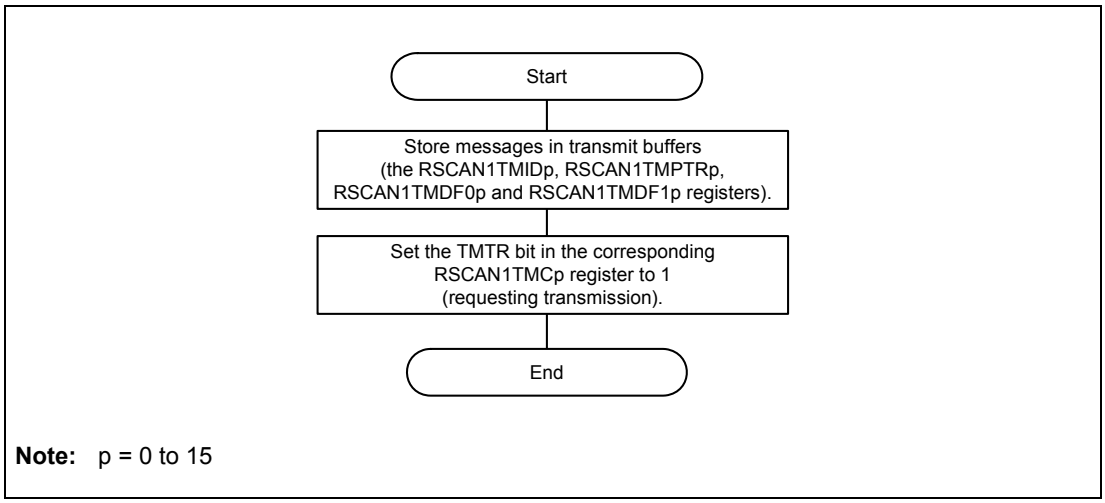
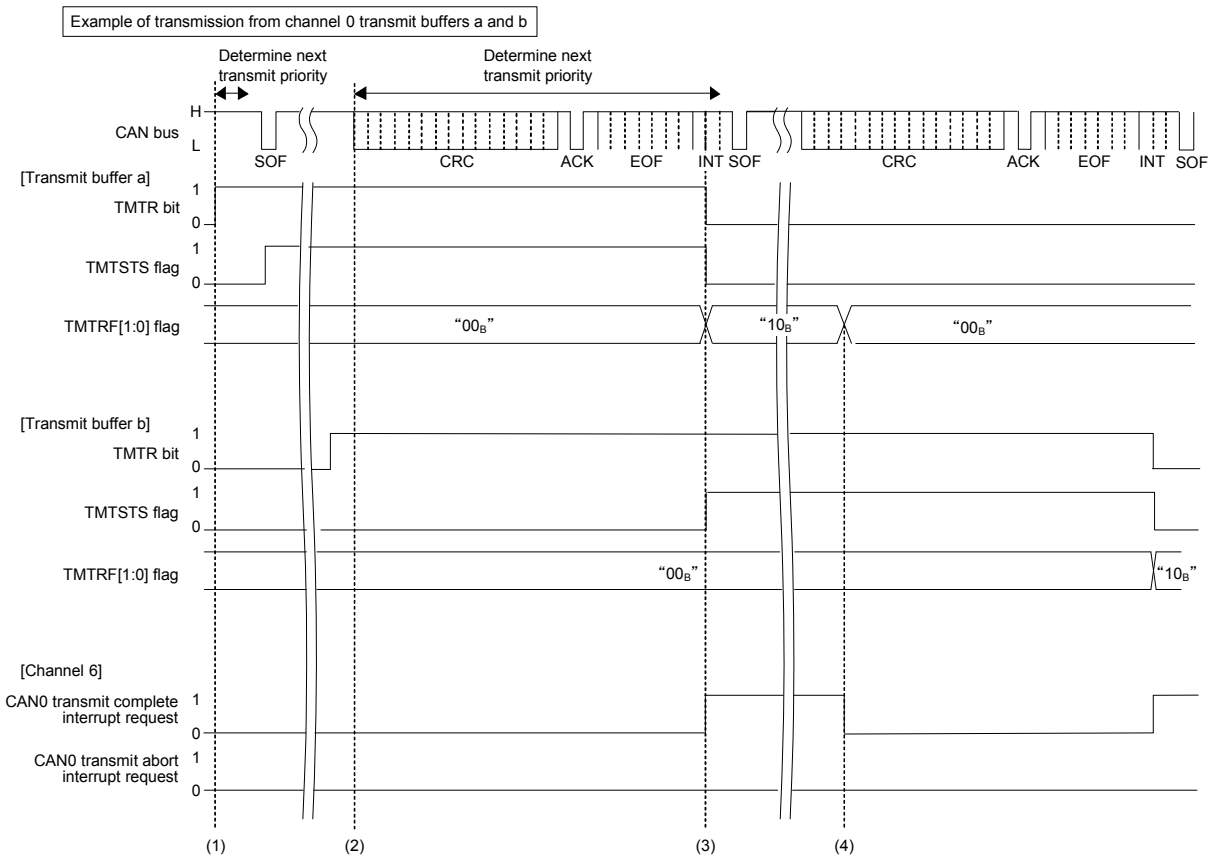



**Figure 21.63 Procedure for Transmission from Transmit Buffers**




**Note:** a = 0 to 15, b = 0 to 15
TMTR: Bit in the RSCAN1TMCp register
TMTSTS, TMTRF[1:0]: Flags in the RSCAN1TMSTSp registers

**Figure 21.64 Transmit Buffer Transmission Timing Chart (Transmission Completed Successfully)**

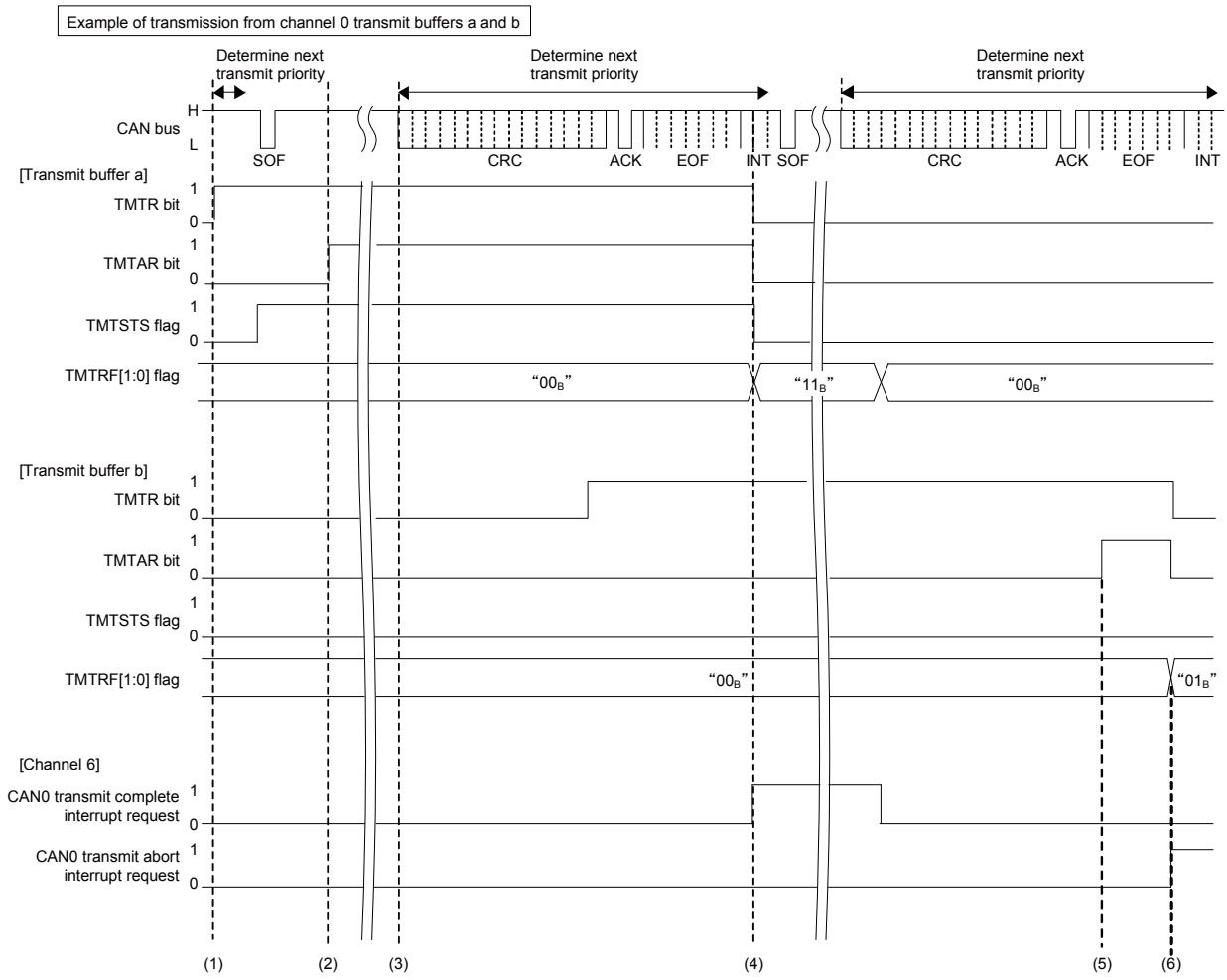
(1) When the TMTR bit in the RSCAN1TMCa register is set to 1 while the CAN bus is idle, the transmit priority determination processing starts to determine the highest-priority transmit buffer. If transmit buffer a is determined to be the highest-priority transmit buffer, the TMTSTS flag in the corresponding RSCAN1TMSTSa register is set to 1 (transmission is in progress) and the CAN channel starts transmitting data.

(2) When a transmit request from a buffer is present, the priority determination starts with the CRC delimiter for the next transmission. The determination time may delay if the transmit priority determination processing is performed on another channel. However, the delay does not occur during transmission because the determination processing is completed by the third bit of the intermission.

(3) When transmission completes successfully, the TMTRF[1:0] flag in the RSCAN1TMSTSa register is set to $10_B$ (transmission has been completed (without transmit abort request)) and the TMTSTS flag and the TMTR bit in the RSCAN1TMCa register are cleared to 0. When the TMIEa bit in the RSCAN1TMIEC0 register is 1 (transmit buffer interrupt is enabled), a CAN0 transmit complete interrupt request is generated. To clear the interrupt request, set the TMTRF[1:0] flag to $00_B$ (transmission is in progress or no transmit request is present).

(4) Before starting the next transmission, set the TMTRF[1:0] flag to $00_B$. Write the next message to the transmit buffer, and then set the TMTR bit to 1 (transmission is requested). The TMTR bit can be set to 1 only when the TMTRF[1:0] flag value is $00_B$.

If an arbitration-lost has occurred after transmission is started, the TMTSTS flag is cleared to 0. The transmit priority determination is reexecuted at the beginning of the CRC delimiter to search the highest-priority transmit buffer. If an error has occurred during transmission or after arbitration loss, the priority determination processing is reexecuted during transmission of an error frame.




**Note:** a = 0 to 15, b = 0 to 15
TMTR, TMTAR: Bits in the RSCAN1TMCp register
TMTSTS, TMTRF[1:0]: Flags in the RSCAN1TMSTSp register

**Figure 21.65 Transmit Buffer Transmission Timing Chart (Transmit Abort Completed)**

(1) When the TMTR bit in the RSCAN1TMCa register is set to 1 while the CAN bus is idle, the transmit priority determination processing starts to determine the highest-priority transmit buffer. If transmit buffer a is determined to be the highest-priority transmit buffer, the TMTSTS flag in the corresponding RSCAN1TMSTSa register is set to 1 (transmission is in progress) and the CAN channel starts transmitting data.

(2)When it is determined that the transmit buffer is used for the next transmission or transmission is in progress, message transmission is not aborted unless an error or arbitration loss occurs even if the TMTAR bit is set to 1 (transmit abort is requested).

(3) The priority determination starts with the CRC delimiter for the next transmission. In this timing chart, buffer b is not selected as the next transmit buffer. The determination time may delay if the transmit priority determination processing is performed on another channel. However, the delay does not occur during transmission because the determination processing is completed by the third bit of the intermission.

(4) When transmission completes successfully, the TMTRF[1:0] flag in the RSCAN1TMSTSa register is set to $11_B$ (transmission has been completed (with transmit abort request)) and the TMTSTS flag and the TMTR bit in the RSCAN1TMCa register are cleared to 0. When the TMIEa value in the RSCAN1TMIEC0 register is 1 (transmit buffer interrupt is enabled), a CAN0 transmit complete interrupt request is generated. To clear the interrupt request, set the TMTRF[1:0] flag to $00_B$ (transmission is in progress or no transmit request is present).

(5) While another CAN node is transmitting data on the CAN bus (TMTSTS flag = 0), if the TMTAR bit is set to 1 while the corresponding channel is determining transmit priority, the TMTR bit cannot be cleared to 0.

(6) After the internal processing time has passed, the transmission is terminated and the TMTRF[1:0] flag is set to $01_B$. When the transmit buffer is not transmitting data and is not selected as the next transmit buffer and priority determination is not being made, an abort request is immediately accepted and the TMTRF[1:0] flag is set to $01_B$. At this time, the TMTR and TMTAR bits are cleared to 0. When transmit abort is completed with the TAIE bit in the RSCAN1CmCTR register set to 1 (transmit abort interrupt is enabled), an interrupt request is generated. To clear the interrupt request, set the TMTRF[1:0] flag to $00_B$.

If an arbitration loss has occurred after the CAN channel started transmission, the TMTSTS bit is cleared to 0. The transmit priority determination is reexecuted at the beginning of the CRC delimiter to find the highest-priority transmit buffer. If an error has occurred during transmission or after arbitration loss, the priority determination processing is reexecuted during transmission of an error frame.

### 21.22.3.2 Procedure for Transmission from Transmit/Receive FIFO Buffers

**Figure 21.66** shows the procedure for transmission from transmit/receive FIFO buffers.

**Figure 21.67** shows a timing chart where messages are transmitted from two transmit/receive FIFO buffers in the same channel and transmission has been successfully completed. **Figure 21.68** shows a timing chart where messages are transmitted from two transmit/receive FIFO buffers in the same channel and transmit abort has been completed.

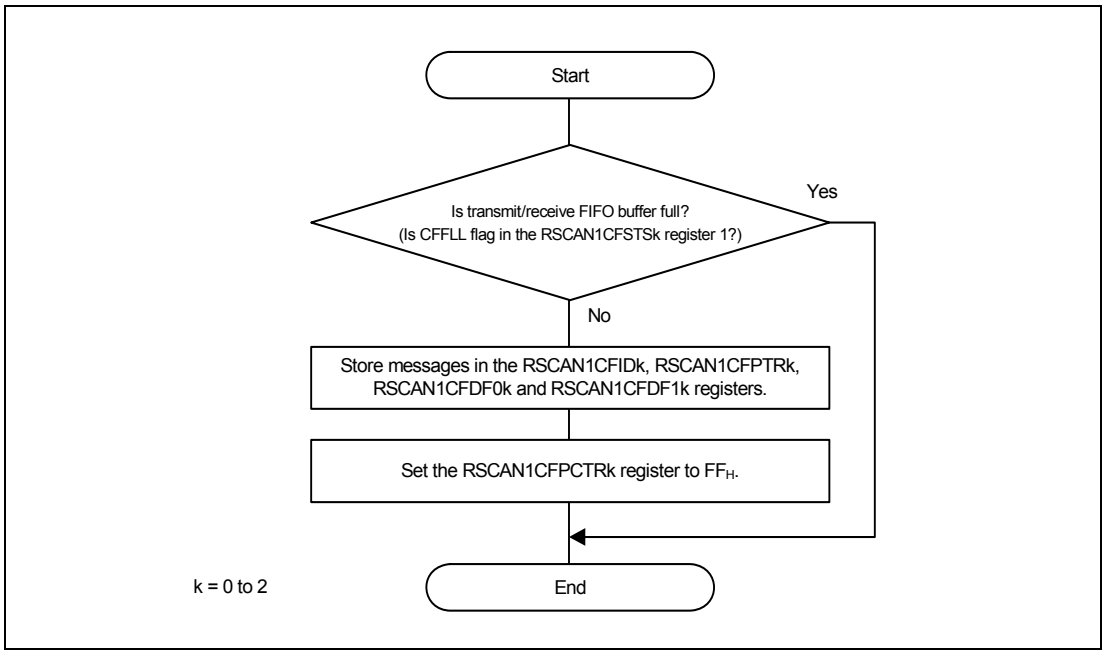



**Figure 21.66 Procedure for Transmission from Transmit/Receive FIFO Buffers**

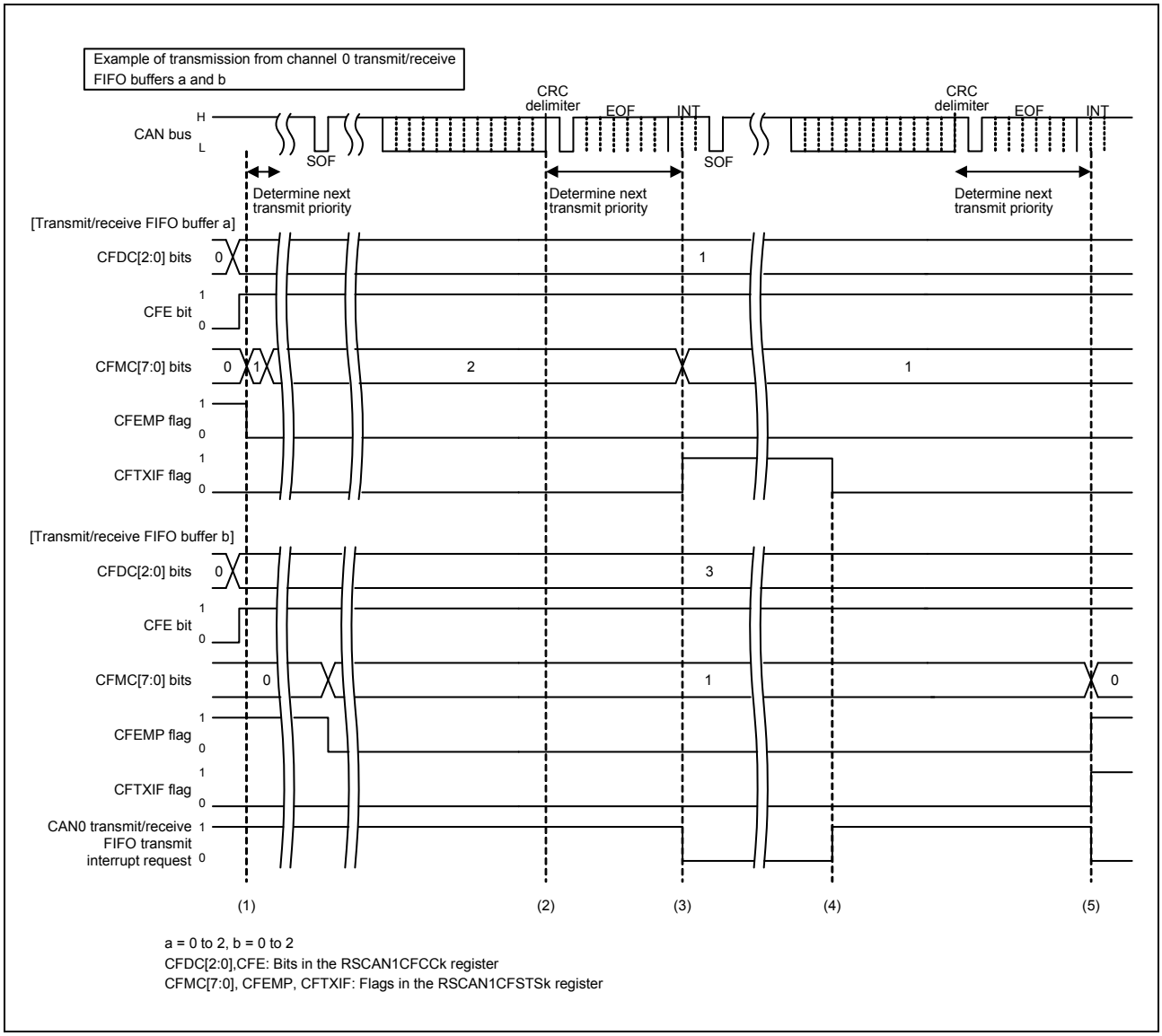



**Figure 21.67 Transmit/Receive FIFO Buffer Transmission Timing Chart (Transmission Completed Successfully)**

(1) While the CAN bus is idle, when the CFE bit in the RSCA0CFCCa register is 1 (transmit/receive FIFO buffers are used) and the CFDC[2:0] value in the RSCAN1CFCCa register is $001_B$ (4 messages) or more and the CFMC[7:0] value in the RSCAN1CFSTSa register is $01_H$ or more, the priority determination processing starts to determine the highest-priority transmit message. When the highest-priority transmit message has been determined, transmission of the message starts. In this figure, the message is transmitted from transmit/receive FIFO buffer a of channel 0.

(2) When a transmit request from a buffer is present, the priority determination starts with the CRC delimiter for the next transmission. The determination time may delay if the transmit priority determination processing is performed on another channel. However, the delay does not occur during transmission because the determination processing is completed by the third bit of the intermission.

(3) When transmission completes successfully, the CFMC[7:0] value in the RSCAN1CFSTSa register is decremented. Setting the CFIM bit in the RSCAN1CFCCa register to 1 (a FIFO transmit

interrupt request is generated each time a message has been transmitted) sets the CFTXIF flag in the RSCAN1CFSTSk register to 1 (a transmit/receive FIFO transmit interrupt request is present).

(4) The program can clear the CFTXIF flag.

(5) Message transmission from transmit/receive FIFO buffer b of channel 0 has been completed and the CFMC[7:0] value in the RSCAN1CFSTSb register is decremented. The CFMC[7:0] bits are cleared to $00_H$ and therefore the CFEMP flag in the RSCAN1CFSTSk register is set to 1 (the transmit/receive FIFO buffer contains no message (buffer empty)).

Transmission is continued until the CFEMP flag is set to 1. It is possible to continuously store transmit messages in FIFO buffers until the CFFLL flag in the RSCAN1CFSTSa and RSCAN1CFSTSb register is set to 1 (the transmit/receive FIFO buffer is full).

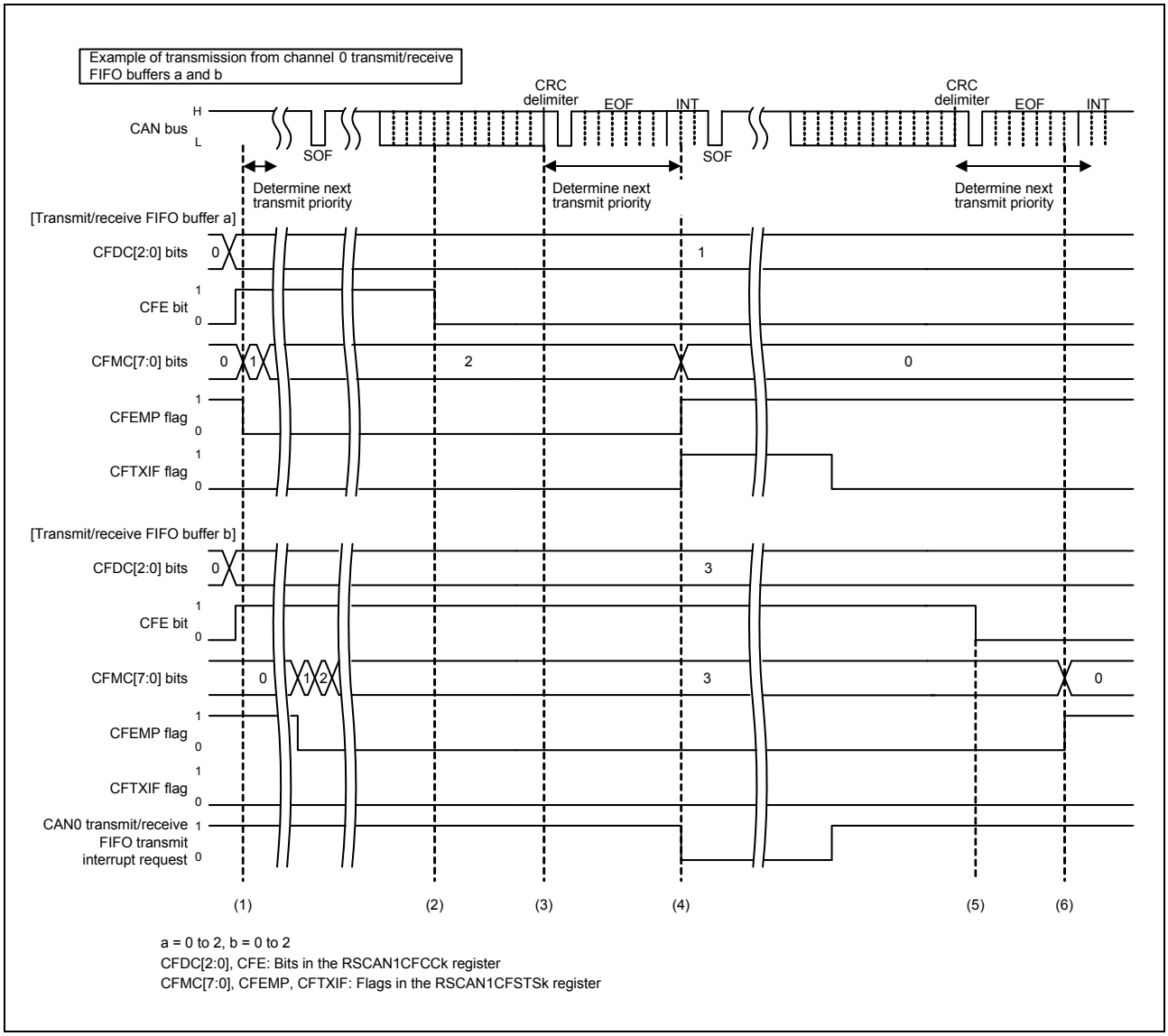



**Figure 21.68 Transmit/Receive FIFO Buffer Transmission Timing Chart (Transmit Abort Completed)**

(1) While the CAN bus is idle, when the CFE bit in the RSCAN1CFCCa register (a = 0 to 17) is 1 (transmit/receive FIFO buffers are used) and the CFDC[2:0] value in the RSCAN1CFCCa register is $001_B$ (4 messages) or more and the CFMC[7:0] value in the RSCAN1CFSTSa register is $01_H$ or more, the priority determination processing starts to determine the highest-priority transmit

message. When the highest-priority transmit message has been determined, transmission of the message starts. In this figure, the message is transmitted from transmit/receive FIFO buffer a of channel 0.

(2) When transmission is in progress or it is determined that the transmit/receive FIFO buffer is used for the next transmission, message transmission is not aborted unless an error or arbitration loss occurs even if the CFE bit is set to 0 (no transmit/receive FIFO buffer is used).

(3) When a transmit request from a buffer is present, the priority determination starts with the CRC delimiter for the next transmission. In this figure, transmit/receive FIFO buffer b is not selected as a buffer for the next transmission. The determination time may delay if the transmit priority determination processing is performed on another channel. However, the delay does not occur during transmission because the determination processing is completed by the third bit of the intermission.

(4) When transmit completes successfully, the CFMC[7:0] value is cleared to $00_H$. Setting the CFIM bit to 1 (a FIFO transmit interrupt request is generated each time a message has been transmitted) sets the CFTXIF flag in the RSCAN1CFSTSa register to 1 (a transmit/receive FIFO transmit interrupt request is present). The program can clear the CFTXIF flag.

(5) If another CAN node on the CAN bus is transmitting data (not from transmit/receive FIFO buffer b), transmit/receive FIFO buffers cannot be disabled immediately even if the CFE bit in the RSCAN1CFCCb register is cleared to 0 (no transmit/receive FIFO buffer is used) during transmit priority determination. (The CFEMP flag in the RSCAN1CFSTSb register is not set to 1 (the transmit/receive FIFO buffer contains no message (buffer empty)) immediately.)

(6) After the internal processing time has passed, transmit/receive FIFO buffers are disabled and the CFMC[7:0] bits in the RSCAN1CFSTSb register are cleared to $00_H$ and the CFEMP flag is set to 1. When the transmit/receive FIFO buffer is not transmitting data and is not selected as the next transmit buffer and priority determination is not in progress, the transmit/receive FIFO buffer is immediately disabled. (The CFMC[7:0] bits are cleared to $00_H$ and the CFEMP flag is set to 1.)

### 21.22.3.3 Procedure for Transmission from the Transmit Queue

**Figure 21.69** shows the procedure for transmission from the transmit queue.

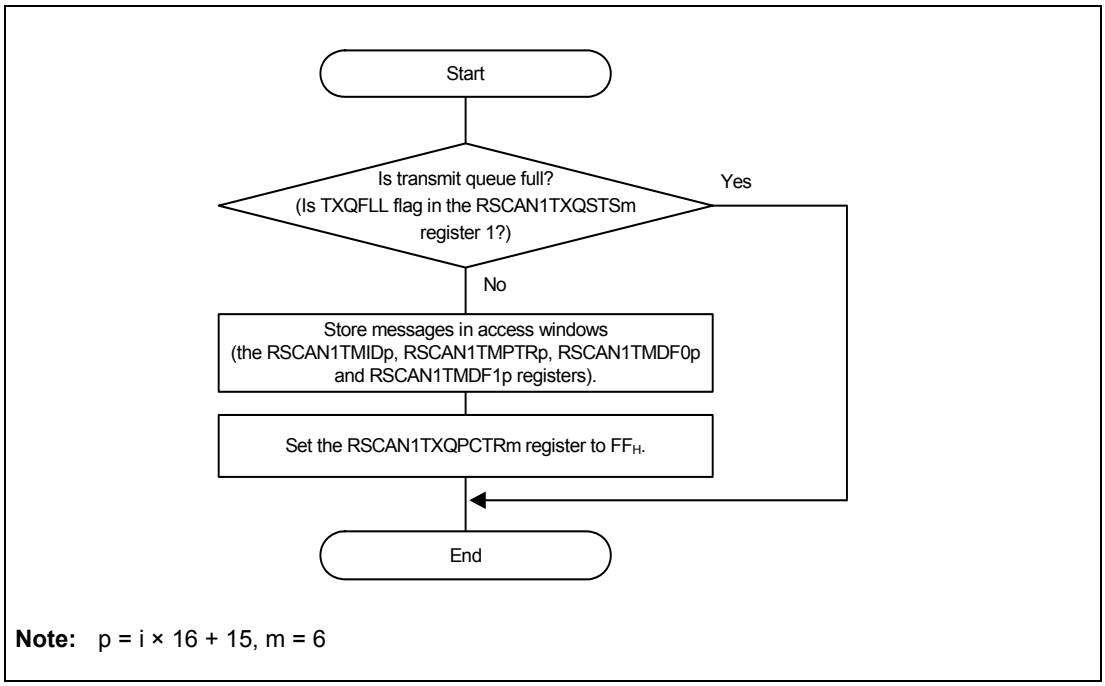
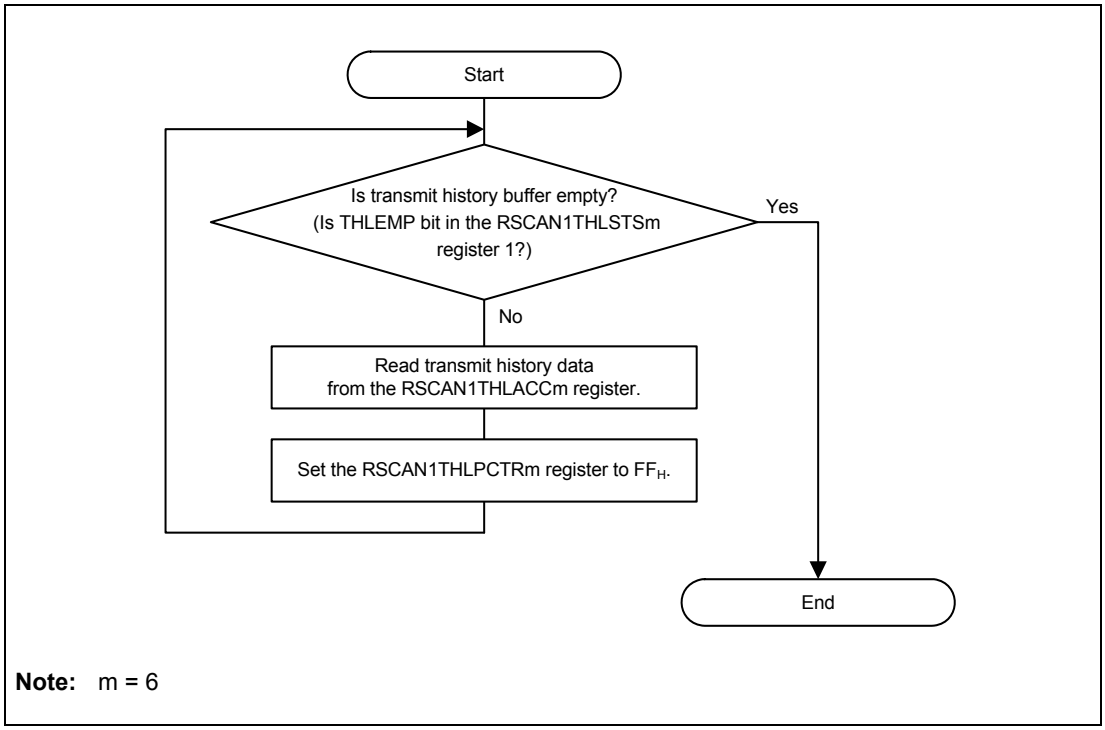



**Figure 21.69 Procedure for Transmission from the Transmit Queue**

### 21.22.3.4 Transmit History Buffer Reading Procedure

Transmit history data can be read from the RSCAN1THLACCm register. The next data can be accessed by writing $FF_H$ to the corresponding RSCAN1THLPCTRm register (m = 6) after reading a set of data. **Figure 21.70** shows the transmit history buffer reading procedure.




**Note:** m = 6

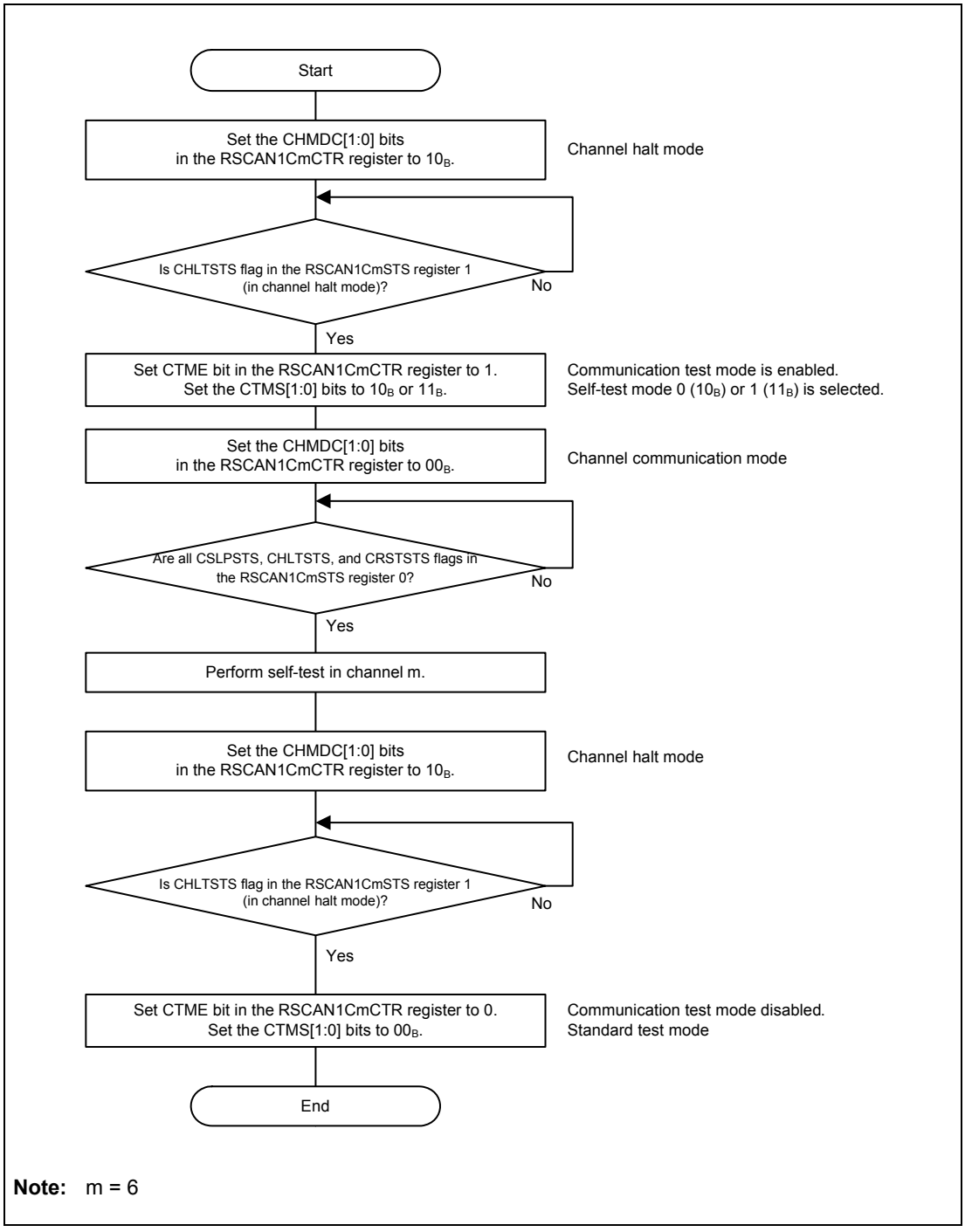
**Figure 21.70 Transmit History Buffer Reading Procedure**

## 21.22.4 Test Settings

### 21.22.4.1 Self-Test Mode Setting Procedure

Self-test mode allows communication test on a channel basis by enabling a CAN node to receive its own transmitted messages.

**Figure 21.71** shows the self-test mode setting procedure.




**Note:** m = 6

**Figure 21.71 Self-Test Mode Setting Procedure**

### 21.22.4.2 Procedure for Releasing the Protection

Since the global test function in **Table 21.297** is protected, write the protection release data 1 and release data 2 in succession to the LOCK[15:0] bits in the RSCAN1GLOCKK register, then set the target test bit to 1.

**Table 21.297 Protection Release Data for Test Function**

| Test Function | Protection Release Data 1 | Protection Release Data 2 | Target Bit |
|---|---|---|---|
| RAM test | $7575_H$ | $8A8A_H$ | RTME bit in the RSCAN1GTSTCTR register |

If an incorrect value is written to the LOCK[15:0] bits, restart from writing the protection release data 1. **Figure 21.72** shows the procedure for releasing the protection.

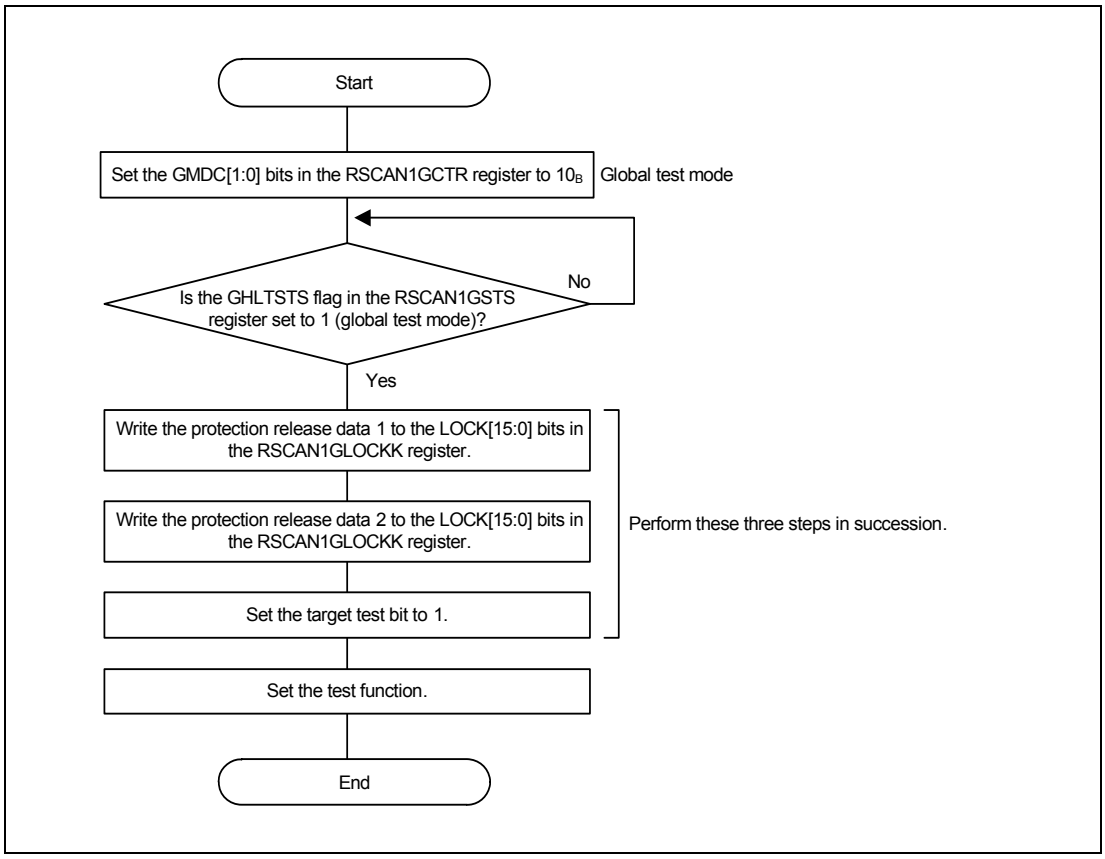



**Figure 21.72 Protection Release Procedure**

### 21.22.4.3 RAM Test Setting Procedure

RAM tests include CAN RAM read/write test. The read/write test verifies that data written to the RAM is read correctly. Before closing the RAM test, write 0000 $0000_H$ to all pages of the CAN RAM.

**Figure 21.73** shows the RAM test setting procedure.

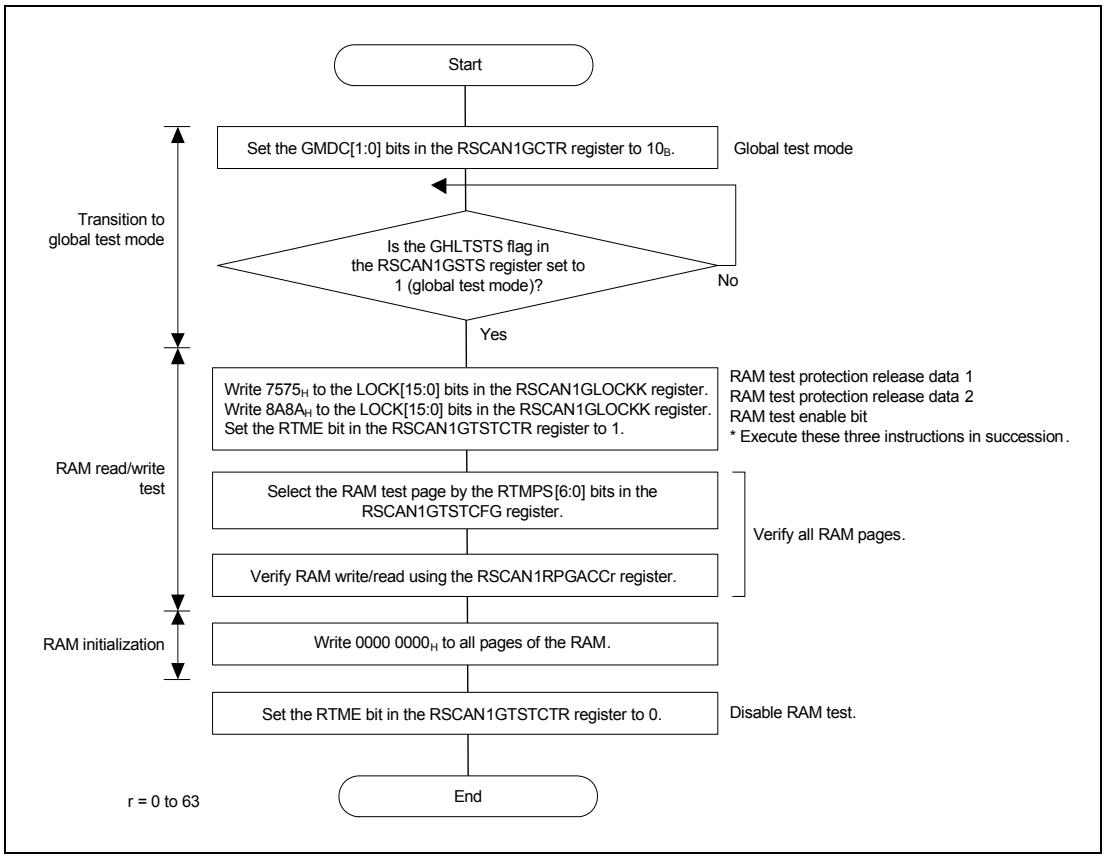



**Figure 21.73 RAM Test Setting Procedure**

# 21.23 Detection and Correction of Errors in RS-CAN RAM

## 21.23.1 ECC for the RSCAN1 RAM

**Table 21.298** gives an outline of the ECC functions for the RSCAN1 RAM.

**Table 21.298 List of the ECC Functions for the RSCAN1 RAM**

| Item | Outline of Functions |
|---|---|
| ECC error detection/ correction | The RAM is checked for ECC errors. The following options can be selected.<br>• 2-bit error detection and 1-bit error detection/correction<br>• 2-bit error detection and 1-bit error detection<br>The ECC error detection/correction can be disabled by using through mode. With the initial settings, error detection/correction is enabled. |
| Error notification | When an ECC error has occurred, the error is notified.<br>• Error notification can be enabled or disabled when an ECC 2-bit error is detected.<br>• Error notification can be enabled or disabled when an ECC 1-bit error is detected.<br>In the initial setting, 2-bit error notification is enabled and 1-bit error notification is disabled. However, when the interrupt is masked by the FEINTFMSK register, interrupt processing is not performed. |
| Error status | Monitoring for the detection of ECC 2-bit errors and for the detection of ECC 1-bit errors is available.<br>A register for clearing the error status is provided. |
| Address capture | Only one address at which an ECC error has occurred can be captured.<br>A signal is generated upon detection of ECC 2-bit or 1-bit error , and the signal is used as a trigger to capture the error-causing address (when the first (1-bit or 2-bit) error is detected after the flag is cleared). |

**CAUTION**

**When ECC error detection/correction is performed, confirm initialization of the RSCAN1 RAM (RSCAN1GSTS.GRAMINIT = 0) before it is used.**

## 21.23.2 Interrupt Request

**Table 21.299** lists the ECC interrupt request of RSCAN1 RAM.

**Table 21.299 RS-CAN ECC Interrupt Request (FE-Level Maskable Interrupt)**

| Unit Interrupt Signal Name | Outline | Name | DMA Trigger Number |
|---|---|---|---|
| ECCDCNRAM1FEIF | RSCAN ECC bit error interrupt | INTECCDCNRAM1 | — |

### 21.23.3 List of Registers

The registers for RSCAN1 are listed in the following table.

**Table 21.300 Registers**

| Module | Register | Abbreviation | Address |
|---|---|---|---|
| ECCCAN1 | RSCAN1 ECC Control Register | ECCRCAN1CTL | FFC7 $1020_H$ |
| ECCCAN1 | RSCAN1 ECC Test Mode Control Register | ECCRCAN1TMC | FFC7 $1024_H$ |
| ECCCAN1 | RSCAN1 ECC Encode/Decode Input/Output Replacement Register | ECCRCAN1TED | FFC7 $102C_H$ |
| ECCCAN1 | RSCAN1 ECC Redundant Bit Data Control Test Register | ECCRCAN1TRC | FFC7 $1028_H$ |
| ECCCAN1 | RSCAN1 ECC Redundant Bit Input/Output Replacement Buffer Register | ECCRCAN1ERDB | FFC7 $1028_H$ |
| ECCCAN1 | RSCAN1 ECC Encode Test Register | ECCRCAN1ECRD | FFC7 $1029_H$ |
| ECCCAN1 | RSCAN1 ECC 7-Bit Redundant Bit Data Hold Test Register | ECCRCAN1HORD | FFC7 $102A_H$ |
| ECCCAN1 | RSCAN1 ECC Decode Syndrome Data Register | ECCRCAN1SYND | FFC7 $102B_H$ |
| ECCCAN1 | RSCAN1 ECC Error Address Register 0 | ECCRCAN1AD0 | FFC7 $1030_H$ |

## 21.23.4 ECCRCANnCTL — RSCAN1 ECC Control Register

The ECCRCANnCTL register controls the mode of the ECC and the status for RSCAN1.

Bits 7, 5 and 4 should be set (written) while the RSCAN1 operation is stopped.

In addition, when writing to bit 7, EMCA1 and EMCA0 need to be $01_B$.

**Access:** This register can be read or written in 16-bit units.

**Address:** See **Table 21.300, Registers**.

**Value after reset:** Undefined

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EMCA1 | EMCA0 | — | — | ECCOVFF | ECER2C | ECER1C | — | ECTHM | — | EC1ECP | EC2EDIC | EC1EDIC | ECER2F | ECER1F | ECEMF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | Undefined |
| R/W | R/W*1 | R/W*1 | R | R | R | R/W*1 | R/W*1 | R | R/W | R | R/W | R/W | R/W | R | R | R |

Note 1. These bits are always read as 0.

**Table 21.301 ECCRCANnCTL Register Contents (1/2)**

| Bit position | Bit Name | Function |
|---|---|---|
| 15 | EMCA1 | Access control bits 1 and 0 to ECC mode selection bit<br>These bits specify whether modifying the ECTHM bit (bit 7) is disabled or enabled. The value written to these bits is not retained. When these bits are read, the read value is always 0. When these bits are $01_B$, writing to bit 7 is enabled. |
| 14 | EMCA0 | |
| 13, 12 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 11 | ECCOVFF | By detecting an error while the error status is set and the new error has anotheraddress than the already latched (not cleared or reset is not issued), this bit is set and error notification is generated. |

| ECCOVFF | Operation explanation |
|---|---|
| 0 | Overflow has not occurred after reset of clearing ECER2F and ECER1F.<br>Clearing at<br>(1) Reset<br>(2) Writing ECER2C = 1 or ECER1C = 1<br>(3) Selecting through mode enable (ECTHM = 1) |
| 1 | Error address register overflowed. |

| | | |
|---|---|---|
| 10 | ECER2C | 2-bit ECC error detection flag clear bit<br>This bit is used to clear the 2 bit error detection flag of ECER2F (bit 2).<br>This bit is always read as 0. Writing 0 is ignored. Write 1 to this bit while the ECER2F bit is set to clear the ECER2F bit. When a conflict between this bit writing and ECER2F bit setting occurs, writing to this bit has a priority. |
| 9 | ECER1C | 1-bit ECC error correction accumulation flag clear bit<br>This bit is used to clear the 1 bit error detection/correction flag of ECER1F (bit 1). This bit is always read as 0. Writing 0 is ignored. Write 1 to this bit while the ECER1F bit is set to clear the ECER1F bit. When a conflict between this bit writing and ECER1F bit setting occurs, writing to this bit has a priority. |
| 8 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 7 | ECTHM | ECC function through mode selection bit<br>This bit is used to set enabling and disabling of ECC.<br><br>Setting this bit to 1 disables ECC function.<br>When writing to this bit, (0, 1) must be written to (EMCA1, EMCA0) at the same time.<br>0: Through mode is disabled (normal operation mode).<br>1: Through mode is enabled. (ECC function disable) |

**Table 21.301 ECCRCANnCTL Register Contents (2/2)**

| Bit position | Bit Name | Function |
|---|---|---|
| 6 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 5 | EC1ECP | 1-bit error correction enable bit<br>This bit specifies whether to enable or disable 1-bit error correction when the ECC error detection/correction is enabled.<br>0: When 1-bit error is detected, the error will be corrected.<br>1: When 1-bit error is detected, the error will not be corrected. |
| 4 | EC2EDIC | 2-bit error detection interrupt control bit<br>This bit controls whether to generate an interrupt when 2-bit error is detected.<br>0: When 2-bit error is detected, a INTECCDCNRAMn interrupt will not be generated.<br>1: When 2-bit error is detected, a INTECCDCNRAMn interrupt will be generated. |
| 3 | EC1EDIC | 1-bit error detection interrupt control bit<br>This bit controls whether to generate an interrupt when 1-bit error is detected.<br>0: When 1-bit error is detected, a INTECCDCNRAMn interrupt will not be generated.<br>1: When 1-bit error is detected, a INTECCDCNRAMn interrupt will be generated. |
| 2 | ECER2F | 2-bit error detection flag bit<br>This flag indicates whether 2-bit error is detected during read access to the RAM when error determination is enabled (ECTHM = 0). When 2-bit error interrupt is enabled (EC2EDIC = 1) and this flag is set, an ECC 2-bit error interrupt (INTECCDCNRAMn) is output. Write 1 to the ECER2C bit (bit 10) to clear the flag. This bit is cleared at the time of through mode enable selection (ECTHM = 1). If 2-bit error is detected again while this bit is set, an interrupt signal will not be generated.<br>0: 2-bit error has not occurred since this bit was cleared.<br>1: 2-bit error has occurred. |
| 1 | ECER1F | 1-bit error detection/correction flag bit<br>This flag indicates whether 1-bit error is detected during read access to the RAM when error determination is enabled (ECTHM = 0). Write 1 to the ECER1C bit (bit 9) to clear the flag. This bit is cleared at the time of through mode enable selection (ECTHM = 1).<br>0: 1-bit error has not occurred since this bit was cleared.<br>1: 1-bit error has occurred. |
| 0 | ECEMF | ECC error message flag<br>This flag indicates whether an error exists in the current read data. This bit is updated whenever the RAM reads data. This bit might be set if it is read before initialization of the RAM. This bit is cleared at the time of through mode enable selection (ECTHM = 1) and when there is no 1 bit error in the decode circuit input data.<br>0: The currently-read RAM data does not have bit errors.<br>1: The currently-read RAM data have bit errors. |

**CAUTION**

Bits 2 and 1 should be cleared when the ECC error message flag (ECEMF) is not set.

We recommend initializing the RAM before clearing bits 2 and 1.

## 21.23.5 ECCRCANnTMC — RSCAN1 ECC Test Mode Control Register

The ECCRCANnTMC register switches to and controls the test mode.

This register can be used when RS-CAN is not accessed to RAM.

**Access:** This register can be read or written in 16-bit units.

**Address:** See **Table 21.300, Registers**.

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ETMA1 | ETMA0 | — | — | — | — | — | — | ECTMCE | — | — | ECTRRS | ECREOS | ECENS | ECDCS | ECREIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W*1 | R/W*1 | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W |

Note 1. These bits are always read as 0.

**Table 21.302 ECCRCANnTMC Register Contents (1/2)**

| Bit position | Bit Name | Function |
|---|---|---|
| 15 | ETMA1 | Access control bits 1 and 0 to ECC test mode bit<br>These two bits specify whether updating the ECTMCE bit (bit 7) is disabled or enabled. The value written to these bits is not retained. When these bits are read, the read value is always 0.<br>When these bits are $10_B$, writing to bit 7 is enabled. |
| 14 | ETMA0 | |
| 13 to 8 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 7 | ECTMCE | ECC test mode enable bit<br>This bit specifies whether to enable access to the test registers and test control bits. When writing to this bit, (1, 0) should be written to (ETMA1, ETMA0) at the same time.<br>0: Access to the test mode registers and bits is disabled.<br>1: Access to the test mode registers and bits is enabled.<br>Test registers: ECCRCANnTED, ECCRCANnTRC, ECCRCANnSYND, ECCRCANnHORD, ECCRCANnECRD, ECCRCANnERDB<br>Register test control bit: ECTRRS, ECREOS, ECENS, ECDCS, ECREIS |
| 6, 5 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 4 | ECTRRS | ECC RAM read test mode selection bit<br>Select reading destination when reading ECCRCANnTED register and reading destination when reading ECCRCANnERDB register.<br>Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously). This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0:The read value of the ECCRCANnTED register is the write value of the ECCRCANnTED register.<br>The read value of the ECCRCANnERDB register is the write value of the ECCRCANnERDB register.<br>1: The read value of the ECCRCANnTED register can read RAM data.<br>The read value of the ECCRCANnERDB register is the ECC data to be written to RAM. |
| 3 | ECREOS | ECC redundant bit output data selection bit<br>This bit is used to identifyand select the ECC data to be stored in RAM as ECC data generated for write data or the value of the ECCRCANnERDB register.<br>Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously).<br>This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: Store ECC data generated for write data to RAM.<br>1: Store the value of ECCRCANnERDB register to RAM. |

**Table 21.302 ECCRCANnTMC Register Contents (2/2)**

| Bit position | Bit Name | Function |
|---|---|---|
| 2 | ECENS | ECC encoder input selection bit<br>This bit is used to identify the targeted data for ECC data generation as write data to RAM or the value of the ECCRCANnTED register.<br>Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously).<br>This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: Generate ECC data from the write data to RAM.<br>1: Generate ECC data from the value of the ECCRCANnTED register. |
| 1 | ECDCS | ECC decoder input selection bit<br>This bit is used to indetifyand select the targeted data for syndrome code generation and error detection as RAM data or the value of the ECCRCANnTED register.<br>Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously). This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: Generate syndrome code from RAM data and detect errors.<br>1: Generate syndrome code from the value of the ECCRCANnTED register and detect errors. |
| 0 | ECREIS | ECC redundant bit input data selection bit<br>This bit is used to indetify and select the targeted data for syndrome code generation and error detection as RAM data or the value of the ECCRCANnERDB register.<br>Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously). This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: Generate syndrome code from ECC data stored in RAM and detect errors.<br>1: Generate syndrome code from the value of the ECCRCANnERDB register and detect errors. |

## 21.23.6 ECCRCANnTED — RSCAN1 ECC Encode/Decode Input/Output Replacement Test Register

In ECC test mode, this register handles test data.

The value of the register can be used to generate ECC data or syndrome code.

When ECC test mode is enabled (ECCRCANnTMC.ECTMCE = 1), it is accessible. When ECCRCANnTMC.ECTMCE = 0, writing to this register is ignored and 0000 $0000_H$ is read.

This register can be used when RS-CAN is not accessed to RAM.

**Access:** This register can be read or written in 32-bit units.

**Address:** See **Table 21.300, Registers**.

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECEDB[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECEDB[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.303 ECCRCANnTED Register Contents**

| Bit position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECEDB[31:0] | When ECCRCANnTMC.ECENS = 1, the value of this register is used to generate ECC data and the value this register to RAM is stored to RAM.<br>When ECCRCANnTMC.ECDCS = 1, the value of this register is used to generate syndrome code and the value of this register is stored in ECC decode syndrome data register (ECCRCANnSYND).<br>In addition, when ECCRCANnTMC.ECTRRS = 1, RAM data [31:0], instead of written data, is read for the value of this register. |

## 21.23.7 ECCRCANnTRC — RSCAN1 ECC Redundant Bit Data Control Test Register

In ECC test mode, this test register, for ECC data, consists of four 8-bit registers, ECCRCANnSYND, ECCRCANnHORD, ECCRCANnECRD, and ECCRSCAN1ERDB.

When ECC test mode is enabled (ECCRCANnTMC.ECTMCE = 1), this register can be accessed. When ECCRCANnTMC.ECTMCE = 0, writing to this register is ignored and 0000 $0000_H$ is read.

This register can be used when RS-CAN is not accessed to RAM.

**Access:** This register can be read or written in 32-bit units.

**Address:** See **Table 21.300, Registers**.

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECCRCANnSYND (See **Section 21.23.8**) | | | | | | | | ECCRCANnHORD (See **Section 21.23.9**) | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECCRCANnECRD (See **Section 21.23.10**) | | | | | | | | ECCRCANnERDB (See **Section 21.23.11**) | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

## 21.23.8 ECCRCANnSYND — RSCAN1 ECC Decode Syndrome Data Register

In ECC test mode, this is a read-only register for storing generated syndrome code.

Writing to this register is ignored.

When ECC test mode is enabled (ECCRCANnTMC.ECTMCE = 1), this register can be accessed. When ECC test mode is disabled (ECCRCANnTMC.ECTMCE = 0), $00_H$ is read.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** See **Table 21.300, Registers**.

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | SYND6 | SYND5 | SYND4 | SYND3 | SYND2 | SYND1 | SYND0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 21.304 ECCRCANnSYND Register Contents**

| Bit position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. |
| 6 to 0 | SYND[6:0] | The generated syndrome code is stored as needed. |

## 21.23.9 ECCRCANnHORD — RSCAN1 ECC 7-Bit Redundant Bit Data Hold Test Register

In ECC test mode, this register is used to store ECC data for read RAM data.

Writing to this register is ignored.

When ECC test mode is enabled (ECCRCANnTMC.ECTMCE = 1), this register can be accessed. When ECC test mode is disabled (ECCRCANnTMC.ECTMCE = 0), $00_H$ is read.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** See **Table 21.300, Registers**.

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | HORD6 | HORD5 | HORD4 | HORD3 | HORD2 | HORD1 | HORD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 21.305 ECCRCANnHORD Register Contents**

| Bit position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. |
| 6 to 0 | HORD[6:0] | ECC code for read RAM data is stored as needed.<br>When ECCRCANnTMC.ECTRRS = 1 and ECCRCANnTED register is read, ECC code is stored. |

## 21.23.10 ECCRCANnECRD — RSCAN1 ECC Encode Test Register

In ECC test mode, this is a read-only register for storing ECC data generated for write RAM data.

Writing to this register is ignored.

When ECC test mode is enabled (ECCRCANnTMC.ECTMCE = 1), this register is accessible. When ECC test mode is disabled (ECCRCANnTMC.ECTMCE = 0), $00_H$ is read.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** See **Table 21.300, Registers**.

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | ECRD6 | ECRD5 | ECRD4 | ECRD3 | ECRD2 | ECRD1 | ECRD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 21.306 ECCRCANnECRD Register Contents**

| Bit position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. |
| 6 to 0 | ECRD[6:0] | These bits can read ECC data generated at the time of RAM data writing, and can read ECC data for the data written to the ECCRCANnTED register when ECCRCANnTMC.ECENS = 1. |

## 21.23.11 ECCRCANnERDB — RSCAN1 ECC Redundant Bit Input/Output Replacement Buffer Register

In ECC test mode, this register handles ECC data.

The value of this register can be used as ECC data generated at the time of RAM writing or ECC data read at the time of RAM data reading.

When ECC test mode is enabled (ECCRCANnTMC.ECTMCE = 1), this register is accessible. When ECC test mode is disabled (ECCRCANnTMC.ECTMCE = 0), $00_H$ is read.

**Access:** This register can be read or written in 8-bit units.

**Address:** See **Table 21.300, Registers**.

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | ERDB6 | ERDB5 | ERDB4 | ERDB3 | ERDB2 | ERDB1 | ERDB0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.307 ECCRCANnERDB Register Contents**

| Bit position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 6 to 0 | ERDB[6:0] | When ECCRCANnTMC.ECREOS = 1, these bits store the value of this register as ECC data to RAM.<br>When ECCRCANnTMC.ECREIS = 1, the value of this register is read as ECC data read from RAM.<br>When ECCRCANnTMC.ECTRRS = 1, ECC data to be stored in RAM is read instead of the write data for the read value of this register. |

## 21.23.12 ECCRCAN1AD0 — RSCAN1 ECC Error Address Register 0

This is read only register to hold the ECC error occurred address.

When ECC error is detected for permitting ECC error judgment, RAM address is captured by the detected signal as trigger and it is hold as the error occurring address.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <ECCCAN1_base > + $10_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | ECEAD[30:16] | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECEAD[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.308 ECCRCAN1AD0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | When read, the value after reset is returned. |
| 30 to 0 | ECEAD[30:0] | ECEAD0 is a read-only register to hold the address at which an ECC error has occurred.<br>If an ECC error is detected while ECC error detection is enabled, the RAM address is latched using the detection signal as a trigger, and the address is stored in ECEAD0 as the address at which the ECC error has occurred.<br>The address is stored upon detection of the first ECC error while no error status is set. However, if a 1-bit error is followed by a 2-bit error, the address of the latter is stored.<br>Only one address can be held in ECEAD0 |

## 21.23.13 SELB_READTEST — ECCREAD Test Select Register

SELB_READTEST is used to check read/write access to the CSIHn ECC registers and the RSCAN1 ECC registers. For detail, see **Section 17.7.14, SELB_READTEST — ECCREAD Test Select Register**.

## 21.24 Notes on the RS-CAN Module

- When changing a global mode, check the GSLPSTS, GHLTSTS, and GRSTSTS flags in the RSCAN1GSTS register for transitions. When changing a channel mode, check the CSLPSTS, CHLTSTS, and CRSTSTS flags in the RSCAN1CmSTS register (m = 6) for transitions.
- The acceptance filter processing checks receive rules sequentially in ascending order from the minimum rule number. If the same ID, IDE bit, or RTR bit value is set for multiple receive rules, the minimum number of receive rule is used for the acceptance filter processing. If the message does not pass through the subsequent DLC filter processing, the data processing is terminated without returning to the acceptance filter processing and the message is not stored in the buffer.
- When linking transmit buffers to transmit/receive FIFO buffers or allocating transmit buffers to transmit queues, set the control register (RSCAN1TMCp) of the corresponding transmit buffer to $00_H$. The status register (RSCAN1TMSTSp) of the corresponding transmit buffer should not be used. Flags in other status registers (registers RSCAN1TMTRSTSy, RSCAN1TMTARSTSy, RSCAN1TMTCSTSy, and RSCAN1TMTASTSy), which correspond to transmit buffers linked to transmit/receive FIFO buffers or allocated to transmit queues remain unchanged. Set the enable bit in the corresponding interrupt enable register (register RSCAN1TMIECy) to 0 (transmit buffer interrupt is disabled).
- Transmit buffers that are linked to transmit/receive FIFO buffers must not be allocated to transmit queues.
- Only a single transmit/receive FIFO buffer can be linked to a transmit buffer. Do not link two or more transmit/receive FIFO buffers to transmit buffers of the same number.
- When the CANm bit time clock is selected as a timestamp counter clock source, the timestamp counter stops when the corresponding channel has transitioned to channel reset mode or channel halt mode.
- In case of an attempt to store a new received message when the receive FIFO buffer and the transmit/receive FIFO buffer are full, the new message is discarded. If you wish to store a new transmit message in the transmit/receive FIFO buffer or the transmit queue, check that the transmit/receive FIFO buffer or the transmit queue is not full.
- The values of unused receive buffers (RSCAN1RMIDq, RSCAN1RMPTRq, RSCAN1RMDF0q, and RSCAN1RMDF1q registers), receive FIFO buffer access registers (RSCAN1RFIDx, RSCAN1RFPTRx, RSCAN1RFDF0x, and RSCAN1RFDF1x registers), and transmit/receive FIFO buffer access registers (RSCAN1CFIDk, RSCAN1CFPTRk, RSCAN1CFDF0k, and RSCAN1CFDF1k registers) are undefined when the RS-CAN module transitions to global operation mode or global test mode after exiting from global reset mode.

# Section 22 Window Watchdog Timer (WDTA)

This section contains a generic description of the Window Watchdog Timer (WDTA).

The first part of this section describes all RH850/F1K specific properties, such as the number of units, register base addresses, etc. The remainder of the section describes the functions and registers of WDTA.

## 22.1 Features of RH850/F1K WDTA

### 22.1.1 Number of Units and Channels

This microcontroller has the following number of WDTA units.

**Table 22.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of Units | 2 | | |
| Name | WDTAn (n = 0, 1) | | |

**Table 22.2 Index**

| Index | Description |
|---|---|
| n | Throughout this section, the individual window watchdog timer units are identified by the index “n”: for example, WDTAnWDTE (n = 0, 1) is the WDTAn enable register. |

### 22.1.2 Register Base Address

WDTAn base addresses are listed in the following table.

WDTAn register addresses are given as offsets from the base addresses.

**Table 22.3 Register Base Addresses**

| Base Address Name | Base Address |
|---|---|
| <WDTA0_base> | FFED $0000_H$ |
| <WDTA1_base> | FFED $1000_H$ |

### 22.1.3 Clock Supply

The WDTAn clock supply is shown in the following table.

**Table 22.4 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| WDTA0 | WDTATCKI | CKSCLK_AWDTA | Timer count clock |
| | Register access clock | CPUCLK2 | Bus clock |
| WDTA1 | WDTATCKI | LS IntOSC | Timer count clock |
| | Register access clock | CPUCLK2 | Bus clock |

### 22.1.4 Interrupt Request

WDTAn interrupt requests are listed in the following table.

**Table 22.5 Interrupt Requests**

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **WDTA0** | | | |
| INTWDTAn | WDTA0 75% interrupt | 40 | — |
| **WDTA1** | | | |
| INTWDTAn | WDTA1 75% interrupt | 41 | — |

**Table 22.6 Interrupt Request (FE Level Non-Maskable Interrupt)**

| Unit Interrupt Signal | Description | Interrupt Name | DMA Trigger Number |
|---|---|---|---|
| **WDTA0** | | | |
| WDTAnTNMI | WDTA0 FENMI interrupt<br>(in WDTA error detection mode with an NMI request) | WDTA0NMI | — |
| **WDTA1** | | | |
| WDTAnTNMI | WDTA1 FENMI interrupt<br>(in the WDTA error detection mode with an NMI request) | WDTA1NMI | — |

### 22.1.5 Reset Sources

WDTAn reset sources are listed in the following table. WDTAn is initialized by these reset sources.

**Table 22.7 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| WDTA0 | Reset sources (AWORES) |
| WDTA1 | All reset sources (ISORES) |

**Note:** WDTA1 is stopped in STOP mode.

## 22.2 Overview

### 22.2.1 Functional Overview

WDTA has the following functions:

- Selection of the operation mode after reset, by using the option bytes

Enabling/disabling of WDTA, starting/stopping of the counter after reset, setting of the counter overflow time, and enabling/disabling of the VAC function can be selected. WDTA startup options to be set by the option bytes are described in **Table 22.8**.

- WDTA trigger function

Writing an activation code to the WDTA trigger register starts WDTA and restarts the counter. Activation codes include fixed activation codes and variable activation codes (VAC function). In a variable activation code, a different value from the previous time (variable value) is written to the WDTA trigger register, which causes the counter to be restarted.

- 75% interrupt request signal

An interrupt request signal can be generated when the WDTA counter reaches 75% of the overflow interval time (this function can be enabled or disabled by the setting of WDTAnMD.WDTAnWIE).

- Window function

The period during which writing to the WDTA trigger register is valid (window-open period) can be set. Writing to the WDTA trigger register during a period other than the window-open period causes an error.

- WDTA error detection function

When an error is detected, a non-maskable interrupt request or an internal reset is generated.

For details about the error sources, see **Section 22.5.3, WDTA Error Detection**.

**Table 22.8 WDTA Start-Up Options**

| Start-Up Option | Function | Description | Option Byte |
|---|---|---|---|
| OPWDEN | WDTA setting | Enables/disables the WDTA.<br>0: WDTA is disabled<br>1: WDTA is enabled | • WDTA0: OPBT0.WDT0_0<br>• WDTA1: OPBT0.WDT1_0 |
| OPWDOVF[2:0] | Overflow interval time reset value setting | Specifies the reset value of the overflow interval time control bits WDTAnMD.WDTAnOVF[2:0]. | • WDTA0/WDTA1: OPBT0.WDT_[2:0] |
| OPWDRUN | Start mode setting | Specifies the start mode.<br>0: Software trigger start mode<br>1: Default start mode<br><br>For details, see **Section 22.5.1, WDTA after Reset Release**. | • WDTA0: OPBT0.WDT0_1<br>• WDTA1: OPBT0.WDT1_1 |
| OPWDVAC | Variable activation code selection | Specifies the trigger register for the generation of counter re-start triggers to keep the counter from overflowing.<br>0: WDTAnWDTE (fixed)<br>1: WDTAnEVAC (variable)<br><br>When WDTAnWDTE is selected, the value to be written to the register (activation code) is fixed (ACH). When WDTAnEVAC is selected, the activation code to be written to the register is variable. For details, see **Section 22.5.2, WDTA Trigger** and **22.5.2.1, Calculating an Activation Code when the VAC Function is Used**. | • WDTA0: OPBT0.WDT0_3<br>• WDTA1: OPBT0.WDT1_3 |

**NOTE**

For the option byte settings, see **Section 37.9, Option Bytes**.

### 22.2.2 Block Diagram

**Figure 22.1** shows the main components of the WDTA.

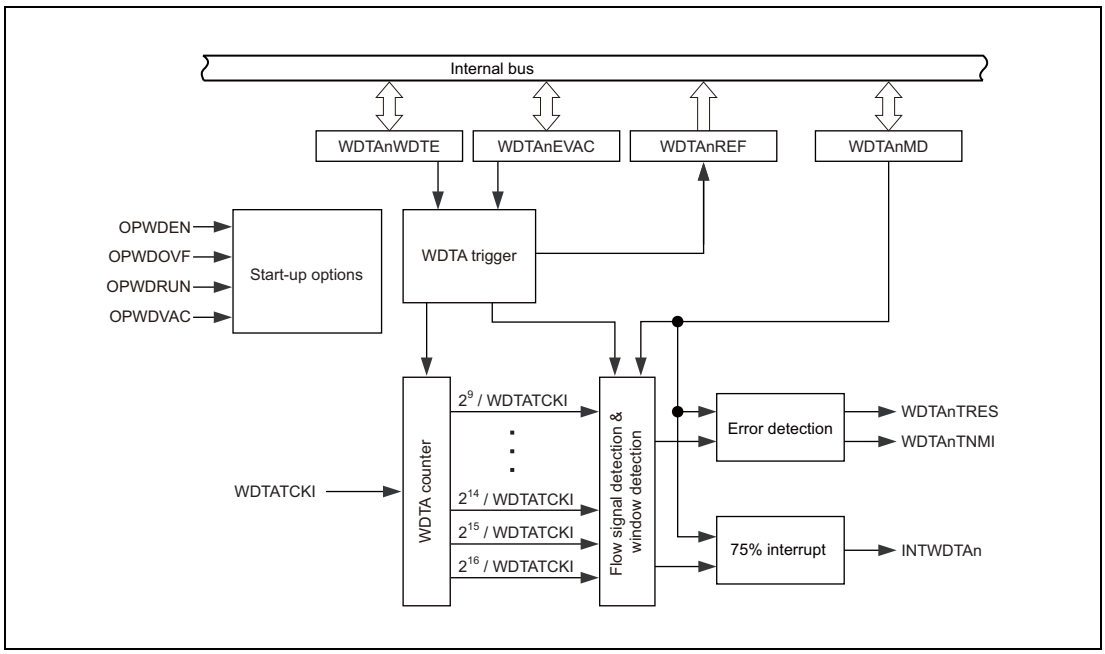



**Figure 22.1 Block Diagram of the Window Watchdog Timer A**

## 22.3 Registers

### 22.3.1 List of Registers

WDTA registers are listed in the following table.

For details about <WDTAn_base>, see **Section 22.1.2, Register Base Address**.

**Table 22.9 List of Registers**

| Module | Register | Symbol | Address |
|---|---|---|---|
| WDTAn | WDTA enable register | WDTAnWDTE | <WDTAn_base> + $0000_H$ |
| | WDTA enable VAC register | WDTAnEVAC | <WDTAn_base> + $0004_H$ |
| | WDTA reference value register | WDTAnREF | <WDTAn_base> + $0008_H$ |
| | WDTA mode register | WDTAnMD | <WDTAn_base> + $000C_H$ |

### 22.3.2 WDTAnWDTE — WDTA Enable Register

This register is the WDTA trigger register when the VAC function is not used (start-up option OPWDVAC = 0).

Writing $AC_H$ to this register generates a WDTA trigger and starts or restarts the WDTA counter. See **Section 22.5.2, WDTA Trigger**, for details.

The behavior of this register depends on the setting of the start-up option (OPWDVAC), see **Table 22.12, WDTAnWDTE Behavior**.

WDTA0 is initialized by AWORES.

WDTA1 is initialized by a reset of any type.

**Access:** This register can be read or written in 8-bit units.

**Address:** <WDTAn_base> + $0000_H$

**Value after reset:** The initial value depends on the start-up options (OPWDEN, OPWDRUN and OPWDVAC). See **Table 22.11, Values of WDTAnRUN7 after Reset**.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WDTAnRUN[7:0] | | | | | | | |
| Value after reset | 0/1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 22.10 WDTAnWDTE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | WDTAnRUN [7:0] | Writing the fixed activation code ($AC_H$) generates the WDTA trigger and starts/ restarts the WDTAn counter. Writing a value other than $AC_H$ generates an error. The WDTAn cannot be stopped once it is started. See **Table 22.12, WDTAnWDTE Behavior**, when reading from or writing to these bits. |

The WDTAnRUN7 bit is only valid if WDTA is enabled (OPWDEN = 1) and the VAC function is disabled (OPWDVAC = 0). **Table 22.11** lists the values of the WDTAnRUN7 bit after reset according to the start-up options.

**Table 22.11 Values of WDTAnRUN7 after Reset**

| Start-Up Options | | | Start Mode | Value of WDTAnRUN7 after Reset |
|---|---|---|---|---|
| OPWDEN | OPWDVAC | OPWDRUN | | |
| 1 | 0 | 1 | Default start | 1 |
| | | 0 | Software trigger start | 0 |

The behavior of WDTAnWDTE during read/write accesses depends on the OPWDVAC setting, as shown in **Table 22.12, WDTAnWDTE Behavior**.

**Table 22.12 WDTAnWDTE Behavior**

| OPWDVAC | Description | WDTAnWDTE | |
|---|---|---|---|
| | | Read | Write |
| 0 | The VAC function is disabled.<br>WDTAnWDTE is enabled. | $2C_H$ is returned (in software trigger start mode, before the activation of WDTAn).<br>$AC_H$ is returned (after the activation of WDTAn). | WDTA trigger<br>Write $AC_H$ *1. |
| 1 | The VAC function is enabled.<br>WDTAnWDTE is disabled. | $2C_H$ is returned. | Writing is ignored. |

Note 1. Any other write value will cause an error.

### 22.3.3 WDTAnEVAC — WDTA Enable VAC Register

This register is the WDTA trigger register when the VAC function is used (start-up option OPWDVAC = 1).

Writing a correct activation code to this register generates a WDTA trigger and starts or restarts the WDTA counter. For details, see **Section 22.5.2, WDTA Trigger**. For details about the activation codes when the VAC function is used, see **Section 22.5.2.1, Calculating an Activation Code when the VAC Function is Used**.

The behavior of this register depends on the setting of the start-up option (OPWDVAC). See **Table 22.15, WDTAnEVAC Behavior**.

WDTA0 is initialized by AWORES.

WDTA1 is initialized by a reset of any type.

**Access:** This register can be read or written in 8-bit units.

**Address:** <WDTAn_base> + $0004_H$

**Value after reset:** The initial value depends on the start-up options (OPWDEN, OPWDRUN and OPWDVAC). See **Table 22.14, Values of WDTAnEVAC7 after Reset**.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WDTAnEVAC[7:0] | | | | | | | |
| Value after reset | 0/1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 22.13 WDTAnEVAC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | WDTAnEVAC [7:0] | Writing a variable activation code generates the WDTA trigger and starts/ restarts the WDTAn counter. Writing an incorrect activation code generates an error. The WDTAn cannot be stopped once it is started.<br>See **Table 22.15, WDTAnEVAC Behavior**, when reading from or writing to these bits. |

The WDTAnEVAC7 bit is only valid if WDTA is enabled (OPWDEN = 1) and the VAC function is enabled (OPWDVAC = 1). **Table 22.14** lists the values of the WDTAnEVAC7 bit after reset according to the start-up options.

**Table 22.14 Values of WDTAnEVAC7 after Reset**

| Start-Up Options | | | Start Mode | Value of WDTAnEVAC7 after Reset |
|---|---|---|---|---|
| OPWDEN | OPWDVAC | OPWDRUN | | |
| 1 | 1 | 1 | Default start | 1 |
| | | 0 | Software trigger start | 0 |

The behavior of WDTAnEVAC during read/write accesses depends on the OPWDVAC setting, as shown in **Table 22.15**.

**Table 22.15 WDTAnEVAC Behavior**

| OPWDVAC | Description | WDTAnEVAC | |
|---|---|---|---|
| | | Read | Write |
| 0 | The VAC function is disabled.<br>WDTAnEVAC is disabled. | $2C_H$ is returned. | Writing is ignored. |
| 1 | The VAC function is enabled.<br>WDTAnEVAC is enabled. | $2C_H$ is returned (in software trigger start mode, before the activation of WDTAn).<br>The variable activation code written last is read (after the activation of WDTAn). | Write the variable activation code*1<br>For details, see **Section 22.5.2.1, Calculating an Activation Code when the VAC Function is Used**. |

Note 1. Any other write value will cause an error.

## 22.3.4 WDTAnREF — WDTA Reference Value Register

This register contains the reference value for calculating the activation code of the VAC function. It is automatically updated after every trigger operation. See **Section 22.5.2.1, Calculating an Activation Code when the VAC Function is Used**.

If the VAC function is disabled (OPWDVAC = 0), reading this register returns $00_H$.

WDTA0 is initialized by AWORES.

WDTA1 is initialized by a reset of any type.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <WDTAn_base> + $0008_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WDTAnREF[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 22.16 WDTAnREF Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | WDTAnREF[7:0] | Reference value for activation code calculation for the VAC function |

### 22.3.5 WDTAnMD — WDTA Mode Register

This register specifies the overflow interval time, 75% interrupt enable/disable, the error mode, and the window-open period.

The value of this register can be updated only once after reset release and before the first trigger is generated. The updated value will be effective after the WDTA trigger register is written to next.

Updating this register after the first WDTA trigger is generated generates an error, but an error does not occur if the same value has been written to it.

WDTA0 is initialized by AWORES.

WDTA1 is initialized by a reset of any type.

**Access:** This register can be read or written in 8-bit units.

**Address:** <WDTAn_base> + $000C_H$

**Value after reset:** The initial value depends on the start-up options (OPWDOVF[2:0]). See **Table 22.8, WDTA Start-Up Options**.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | WDTAnOVF[2:0] | | | WDTAnWIE | WDTAnERM | WDTAnWS[1:0] | |
| Value after reset | 0 | *1 | *1 | *1 | 0 | 1 | 1 | 1 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The WDTAnOVF[2:0] value after reset can be set by the start-up options OPWDOVF[2:0].

**Table 22.17 WDTAnMD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 6 to 4 | WDTAnOVF[2:0] | Selects the overflow interval time |

| WDTAn OVF2 | WDTAn OVF1 | WDTAn OVF0 | Overflow Interval Time |
|---|---|---|---|
| 0 | 0 | 0 | $2^9$ / WDTATCKI |
| 0 | 0 | 1 | $2^{10}$ / WDTATCKI |
| 0 | 1 | 0 | $2^{11}$ / WDTATCKI |
| 0 | 1 | 1 | $2^{12}$ / WDTATCKI |
| 1 | 0 | 0 | $2^{13}$ / WDTATCKI |
| 1 | 0 | 1 | $2^{14}$ / WDTATCKI |
| 1 | 1 | 0 | $2^{15}$ / WDTATCKI |
| 1 | 1 | 1 | $2^{16}$ / WDTATCKI |

| Bit Position | Bit Name | Function |
|---|---|---|
| 3 | WDTAnWIE | Enables/disables the 75% interrupt request INTWDTAn.<br>0: INTWDTAn is disabled.<br>1: INTWDTAn is enabled. |
| 2 | WDTAnERM | Specifies the error mode.<br>0: NMI request mode<br>1: Reset mode |
| 1, 0 | WDTAnWS[1:0] | Selects the window-open period. |

| WDTAn WS1 | WDTAn WS0 | Window-Open Period |
|---|---|---|
| 0 | 0 | 25% |
| 0 | 1 | 50% |
| 1 | 0 | 75% |
| 1 | 1 | 100% |

## 22.4 Interrupt Sources

WDTA checks the status of the WDTA counter value, detects illegal accesses to the WDTA-related registers, and generates an interrupt request. The following are WDTA interrupt requests:

(1) INTWDTAn (WDTA timer count 75% interrupt request)
An interrupt request signal is generated at 75% of the counter overflow time of the WDTA timer. An interrupt request signal can be enabled or disabled by using the WDTA mode register (WDTAnMD).

(2) WDTAnTNMI (WDTA error detection interrupt)
Detection of a WDTA error to generation of an NMI interrupt request. The WDTA mode register (WDTAnMD) can be used to switch between an NMI interrupt and a reset. For details about WDTA errors, see **Section 22.5.3, WDTA Error Detection**.

# 22.5 Functions

## 22.5.1 WDTA after Reset Release

### 22.5.1.1 Start Modes

There are two start modes (software start mode and default start mode) when WDTAn starts after reset release. The start mode can be selected by the start-up option.

The start mode selection is listed in **Table 22.18**.

**Table 22.18 Start Modes**

| Start-Up Option OPWDRUN | Start Mode | Description |
|---|---|---|
| 0 | Software trigger | • The WDTA counter stops ($0000_H$) after reset release.<br>• Writing an activation code to the WDTA trigger register starts WDTA. |
| 1 | Default | The WDTA counter starts after reset release. |

### 22.5.1.2 WDTA Settings after Reset Release

(1) **Table 22.19** shows the WDTA settings after reset release.

**Table 22.19 WDTA Settings after Reset Release**

| Function | Setting | Remark |
|---|---|---|
| WDTA enable/disable | Specified by start-up options | |
| Start mode | | |
| VAC function | | |
| WDTA overflow interval time | Specified by start-up options | Modification is possible only once by the setting of the WDTA mode register (WDTAnMD). |
| 75% interrupt mode | 75% interrupt disabled | |
| Behavior on error detection | Reset generation | |
| Window-open period | 100% | |

The setting of the WDTA mode register (WDTAnMD) is enabled when the first WDTA trigger is generated (writing an activation code to WDTAnWDTE and WDTAnEVAC). Perform the WDTAnMD register setting before a WDTA trigger is generated.

Setting of WDTA by using WDTAnMD is possible only once. If the value set to WDTAnMD is changed after a WDTA trigger is generated, an error occurs. However, an error does not occur if the same value is set.

### 22.5.1.3 Default Start Mode Timing

The default start mode timing and the changes to the WDTA settings are illustrated in **Figure 22.2**.

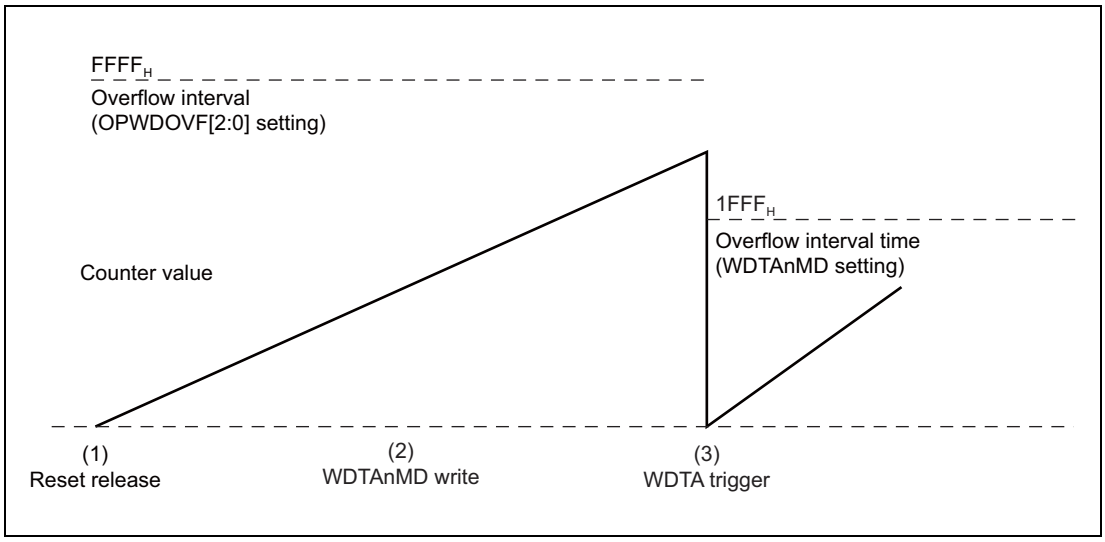



**Figure 22.2 Timing Diagram of WDTA Start in Default Start Mode**

The timing diagram shown in **Figure 22.2** shows the following behaviors:

(1) In default start mode, the WDTA counter starts after reset release. The overflow interval time after reset release is set by start-up options.

Example: Overflow interval time after reset release
$= 2^{16}$/WDTATCKI (OPWDOVF[2:0] = $111_B$)

(2) WDTAnMD is set before a WDTA trigger is generated. Note that the setting is not applied immediately.

(3) Write to the WDTA trigger register before the WDTA counter overflows.
The WDTAnMD setting is applied by the generation of a WDTA trigger.

Example: Overflow interval time after a WDTA trigger is generated
$= 2^{13}$/WDTATCKI

### 22.5.1.4 Software Trigger Start Mode Timing

The software trigger start mode timing and the changes to the WDTA settings are illustrated in **Figure 22.3**.

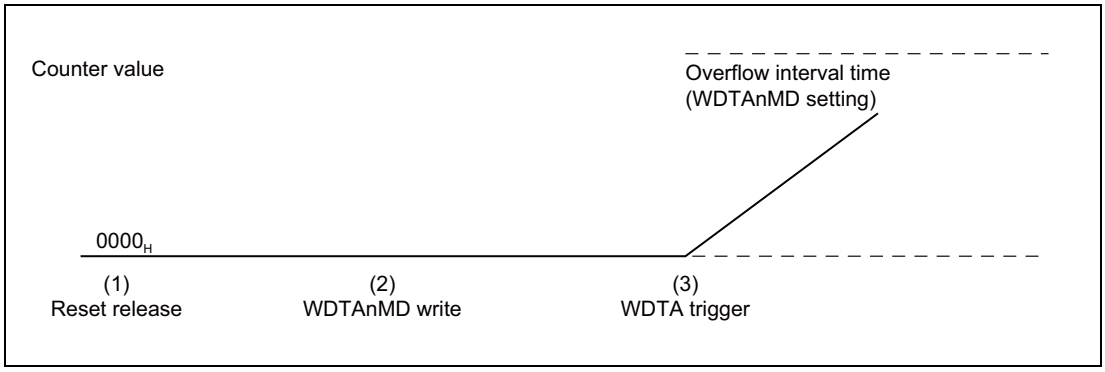



**Figure 22.3 Timing Diagram of WDTA Start in Software Trigger Start Mode**

The timing diagram shown in **Figure 22.3** shows the following behaviors:

(1) The WDTA counter value remains $0000_H$ until the first trigger is generated after reset release. The overflow interval time is set by start-up options, but this setting has no effect because the counter is not operating.

(2) WDTAnMD is set before a WDTA trigger is generated. Note that the setting is not applied immediately.

(3) The WDTA counter starts by the generation of a WDTA trigger.
The overflow interval time specified in WDTAnMD and other settings are applied.

### 22.5.2 WDTA Trigger

Writing a special value called an activation code to the WDTA enable register (WDTAnWDTE) and the WDTA enable VAC register (WDTAnEVAC) leads to generation of a WDTA trigger.

The WDTA trigger has the following functions:

- Starting the WDTA counter in software trigger start mode
- Restarting the WDTA counter
- Setting the WDTA mode specified by the WDTAnMD register (only for the first WDTA trigger after reset release)

The WDTA trigger register, which generates a WDTA trigger, is specified by the start-up option OPWDVAC.

**Table 22.20** lists the WDTA trigger registers and activation codes.

**Table 22.20 WDTA Trigger Registers and Activation Codes**

| Type of Activation Code | Trigger Register | Activation Code |
|---|---|---|
| Fixed (OPWDVAC = 0) | WDTAnWDTE | $AC_H$ |
| Variable (OPWDVAC = 1) | WDTAnEVAC | For details, see **22.5.2.1, Calculating an Activation Code when the VAC Function is Used**. |

**CAUTION**

- **When writing the processing to clear WDTA successively, the clear processing below is not acknowledged for the following period:**
  **"12 × CPU clock*1 cycles + 6 × WDT clock*2 cycles"**
- **After writing the processing to clear WDTA and then changing to standby mode, the clear processing below is not acknowledged for the following period after return from stand-by mode:**
  **"6 × CPU clock*1 cycles + 3 × WDT clock*2 cycles"**

Note 1. CPU clock: Clock selected by CKSC_CPUCLKS_CTL and CKSC_CPUCLKD_CTL
Note 2. WDT clock: Clock selected by CKSC_AWDTAD_CTL in WDTA0.
LS IntOSC in WDTA1.

#### 22.5.2.1 Calculating an Activation Code when the VAC Function is Used

Use the following expression to calculate the variable activation code (ExpectWDTE) to be set in the WDTA trigger register (WDTAnEVAC) when the VAC function is used, by using the WDTA reference value register (WDTAnREF):

$$\text{ExpectWDTE} = AC_H - \text{WDTAnREF (previous)}$$

Note that the value in the WDTAnREF register is updated every time a start-code is written to the trigger register WDTAnEVAC. Use the following expression to calculate the updated value of the WDTAnREF register:

WDTAnREF (following) = (rotate the value of ExpectWDTE to the left by 1 bit)

**Table 22.21** lists the variable activation codes according to the number of WDTA triggers.

**Table 22.21 Expected Variable Activation Code Development**

| No*1 | WDTAnREF (Previous) | | ExpectWDTE ($AC_H$ - WDTAnREF) | | WDTAnREF (Following) | |
|---|---|---|---|---|---|---|
| 0 | 0000 0000 | $00_H$ | 1010 1100 | $AC_H$ | 0101 1001 | $59_H$ |
| 1 | 0101 1001 | $59_H$ | 0101 0011 | $53_H$ | 1010 0110 | $A6_H$ |
| 2 | 1010 0110 | $A6_H$ | 0000 0110 | $06_H$ | 0000 1100 | $0C_H$ |
| … | … | … | … | … | … | … |

Note 1. Number of triggers after reset

**NOTE**

Writing an incorrect activation code generates an error.

### 22.5.3 WDTA Error Detection

WDTA detects an error, including generation of the WDTA count overflow or illegal operations.

The following events are detected as errors:

- WDTA counter overflow
- Incorrect activation code is written to the WDTA trigger register
- Writing to the trigger register at a time outside the window-open period
- When the setting value in the WDTA mode register (WDTAnMD) is changed after the first WDTA trigger is generated
- When the value of the WDTA mode register (WDTAnMD) is changed twice before the first WDTA trigger is generated

#### 22.5.3.1 WDTA Error Mode

When a WDTA error is detected, either an NMI interrupt or a reset is generated according to the setting of the WDTA error mode bit (WDTAnMD.WDTAnERM). The error mode bit after reset release is set to the reset mode.

- WDTAnMD.WDTAnERM = 0: NMI mode
- WDTAnMD.WDTAnERM = 1: Reset mode

**Figure 22.4** shows the reset or NMI request generation when the counter overflows and default start mode is selected.

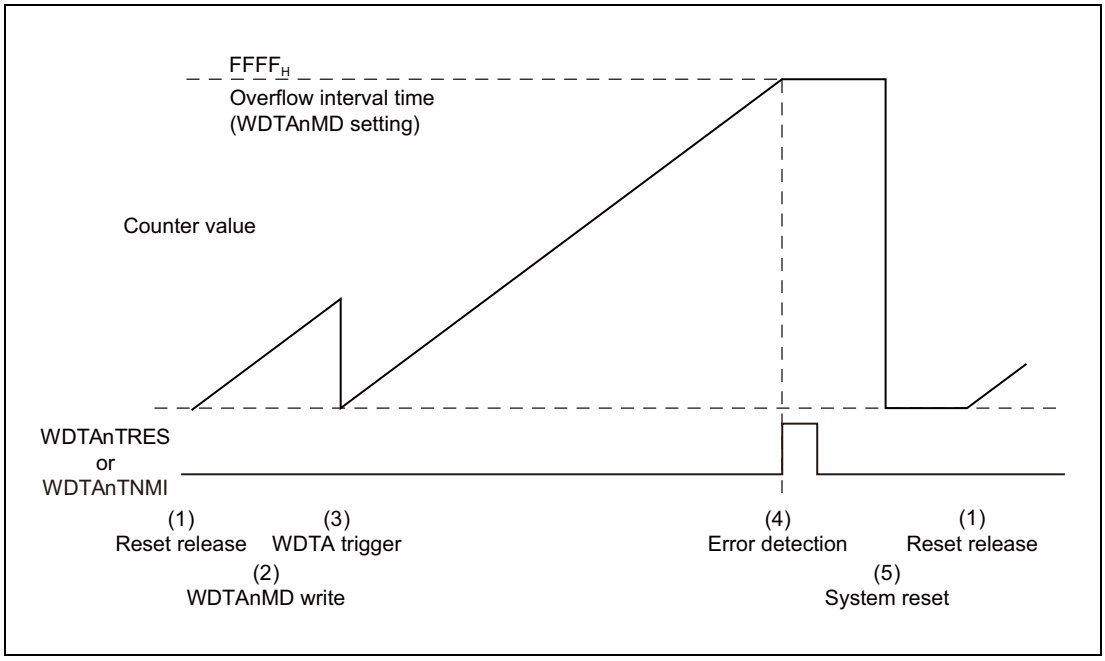



**Figure 22.4 Timing Diagram of WDTA NMI Request or Reset Generation**

The timing diagram shown in **Figure 22.4** shows the following behaviors:

(1) In default start mode, the WDTA counter starts after reset release. The overflow interval time after reset release is set by start-up options.

(2) WDTAnMD is set before a WDTA trigger is generated.
In this case, $2^{16}$/WDTATCKI is set for the overflow interval time.

(3) A WDTA trigger is generated, and the WDTAnMD setting is applied.

(4) When the counter overflows, an error is detected. Depending on the error mode, either interrupt request WDTAnTNMI or reset WDTAnTRES is generated.
The counter value remains until a system reset is performed.

(5) When the system is reset, the counter is cleared and stops until reset release.

### 22.5.4 75% Interrupt Request Signals

When the WDTA counter reaches 75% of the time set for the overflow interval, the interrupt request INTWDTAn is generated.

This function can be enabled or disabled by the WDTAnMD.WDTAnWIE register.

**Figure 22.5** shows the 75% interrupt request generated under following conditions:

- Default start mode is selected.
- 75% interrupt request is enabled after the first WDTA trigger is generated
- WDTA overflow interval time is $2^{16}$/WDTATCKI

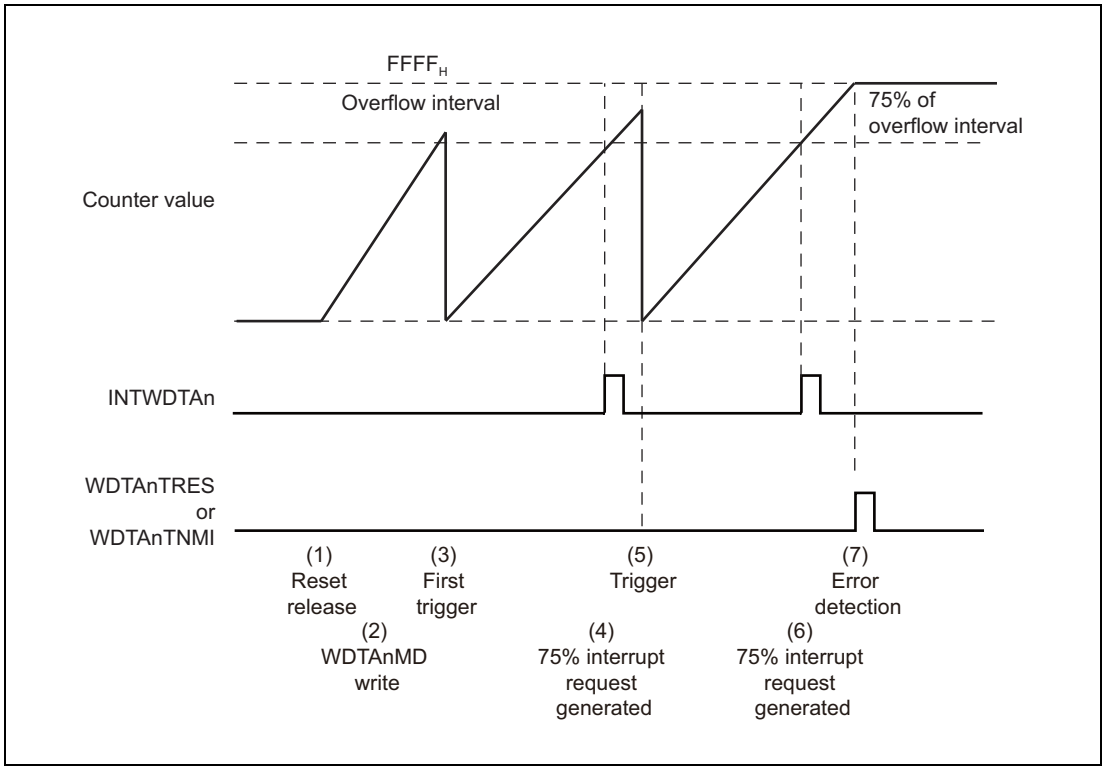



**Figure 22.5 Timing Diagram of WDTA 75% Interrupt Request Signals**

(1) In default start mode, the WDTA counter starts after reset release. The overflow interval time after reset release is set by start-up options.

(2) WDTAnMD is set before a WDTA trigger is generated. In this case, $2^{16}$/WDTATCKI is set for the overflow interval time.

(3) A WDTA trigger is generated, and the WDTAnMD setting is applied.

(4) When the WDTA counter reaches 75% of the overflow interval timer, interrupt request INTWDTAn is generated.

(5) The WDTA trigger restarts counting.

(6) When the WDTA counter reaches 75% of the overflow interval timer, interrupt request INTWDTAn is generated.

(7) When the counter overflows, an error is detected. Depending on the error mode, either interrupt request WDTAnTNMI or reset WDTAnTRES is generated. The counter value remains until a system reset is performed.

### 22.5.5 Window Function

The period when a WDTA trigger is valid (window-open period) can be set. If the window-open period is set to the value less than 100%, the generation of a WDTA trigger not in the window-open period causes an error. The window-open period after reset release is 100%. The period is set to the value by the WDTAnMD.WDTAnWS[1:0] setting after the first WDTA trigger is generated.

**Figure 22.6** shows the behavior of the window function under the following conditions.

- Default start mode is selected.
- 25% window-open period is enabled after the first WDTA trigger is generated (WDTAnWS[1:0] = $00_B$)
- WDTA overflow interval time is $2^{16}$/WDTATCKI

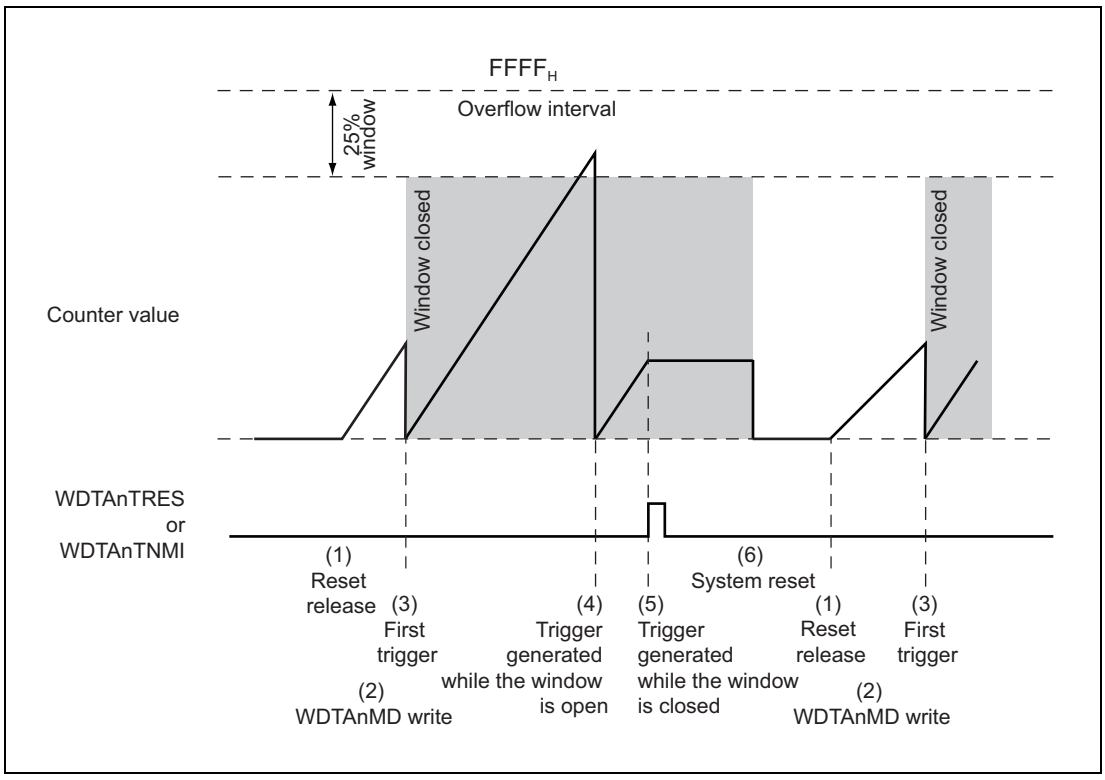



**Figure 22.6 Timing Diagram of WDTA Window Function**

(1) In default start mode, the WDTA counter starts after reset release. The overflow interval time after reset release is set by start-up options.

(2) WDTAnMD is set before a WDTA trigger is generated. In this case, $2^{16}$/WDTATCKI is set for the overflow interval time.

(3) A WDTA trigger is generated, and the WDTAnMD setting is applied.

(4) During the window-open period, the WDTA trigger restarts counting.

(5) During the window-closed period, an error is detected by the WDTA trigger. Depending on the error mode, either interrupt request WDTAnTNMI or reset WDTAnTRES is generated. The counter value remains until a system reset is performed.

(6) When the system is reset, the counter is cleared and stops until reset release.

# Section 23 OS Timer (OSTM)

This section contains a generic description of the OS Timer (OSTM).

The first part of this section describes all RH850/F1K specific properties, such as the number of units, register base addresses, etc. The remainder of the section describes the functions and registers of the OSTM.

## 23.1 Features of RH850/F1K OSTM

### 23.1.1 Number of Units

This microcontroller has the following number of units of the OSTM.

**Table 23.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of Units | 5 | | |
| Name | OSTMn (n = 0 to 4) | | |

**Table 23.2 Index**

| Index | Description |
|---|---|
| n | Throughout this section, the individual OSTM units are identified by the index “n”; for example, OSTMnCNT is the OSTM counter register. |

### 23.1.2 Register Base Address

OSTM base addresses are listed in the following table.

OSTM register addresses are given as offsets from the base addresses.

**Table 23.3 Register Base Addresses**

| Base Address Name | Base Address |
|---|---|
| <OSTM0_base> | FFD7 $0000_H$ |
| <OSTM1_base> | FFD7 $0100_H$ |
| <OSTM2_base> | FFD7 $0200_H$ |
| <OSTM3_base> | FFD7 $0300_H$ |
| <OSTM4_base> | FFD7 $0400_H$ |

### 23.1.3 Clock Supply

The OSTM clock supply is shown in the following table.

**Table 23.4 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| OSTMn (n = 0 to 4) | PCLK | CPUCLK2 | Timer count clock |
| | Register access clock | CPUCLK2 | Bus clock |

### 23.1.4 Interrupt Request

OSTM interrupt requests are listed in the following table.

**Table 23.5 Interrupt Requests**

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **OSTM0** | | | |
| OSTMTINT | OSTM0 interrupt | 84 (INTOSTM0) | — |

**Table 23.6 Interrupt Request (FE Level Maskable Interrupt Request)**

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **OSTM0** | | | |
| OSTMTINT | OSTM0 interrupt | INTOSTM0_FE | — |
| **OSTM1** | | | |
| OSTMTINT | OSTM1 interrupt | INTOSTM1_FE | — |
| **OSTM2** | | | |
| OSTMTINT | OSTM2 interrupt | INTOSTM2_FE | — |
| **OSTM3** | | | |
| OSTMTINT | OSTM3 interrupt | INTOSTM3_FE | — |
| **OSTM4** | | | |
| OSTMTINT | OSTM4 interrupt | INTOSTM4_FE | — |

### 23.1.5 Reset Sources

OSTM reset sources are listed in the following table. OSTM is initialized by these reset sources.

**Table 23.7 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| OSTMn (n = 0 to 4) | All reset sources (ISORES) |

## 23.2 Overview

OSTM is a 32-bit timer/counter.

It can be used in interval timer mode or in free-run compare mode. The settings for operating mode specify the direction of counting (up or down) to control the generation of interrupt requests.

### 23.2.1 Functional Overview

OSTM has the following features.

- Two operating modes
  - Interval timer mode
  - Free-run compare mode
- OSTMTINT interrupt

### 23.2.2 Block Diagram

The following block diagram shows the main components of OSTM.

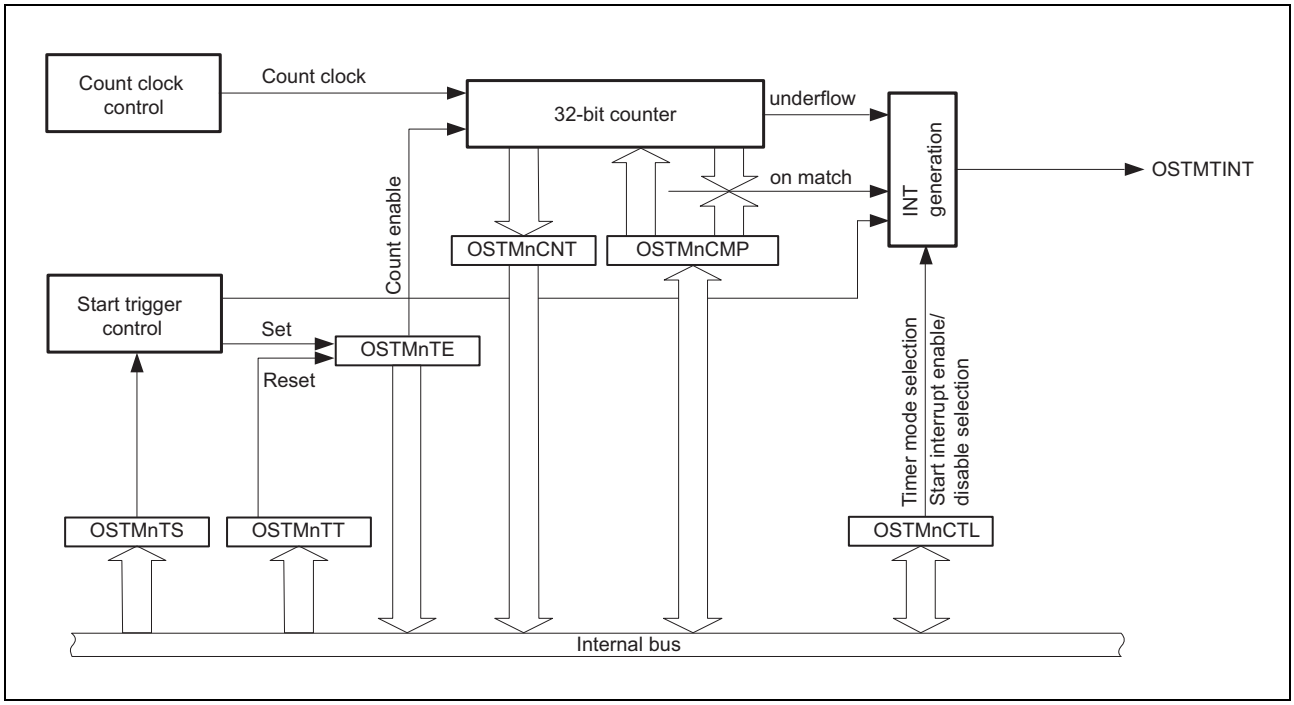



**Figure 23.1 Block Diagram of OSTM**

### 23.2.3 Count Clock

The count clock used by OSTM is PCLK.

### 23.2.4 Interrupt Sources (OSTMTINT)

An OSTMTINT interrupt request is generated on counter underflow (interval timer mode) or when the counter matches the compare value (free-run compare mode).

An interrupt request can also be generated on starting and restarting of the counter. This is controlled by the OSTMnCTL.OSTMnMD0 bit.

This is illustrated in the following figure.

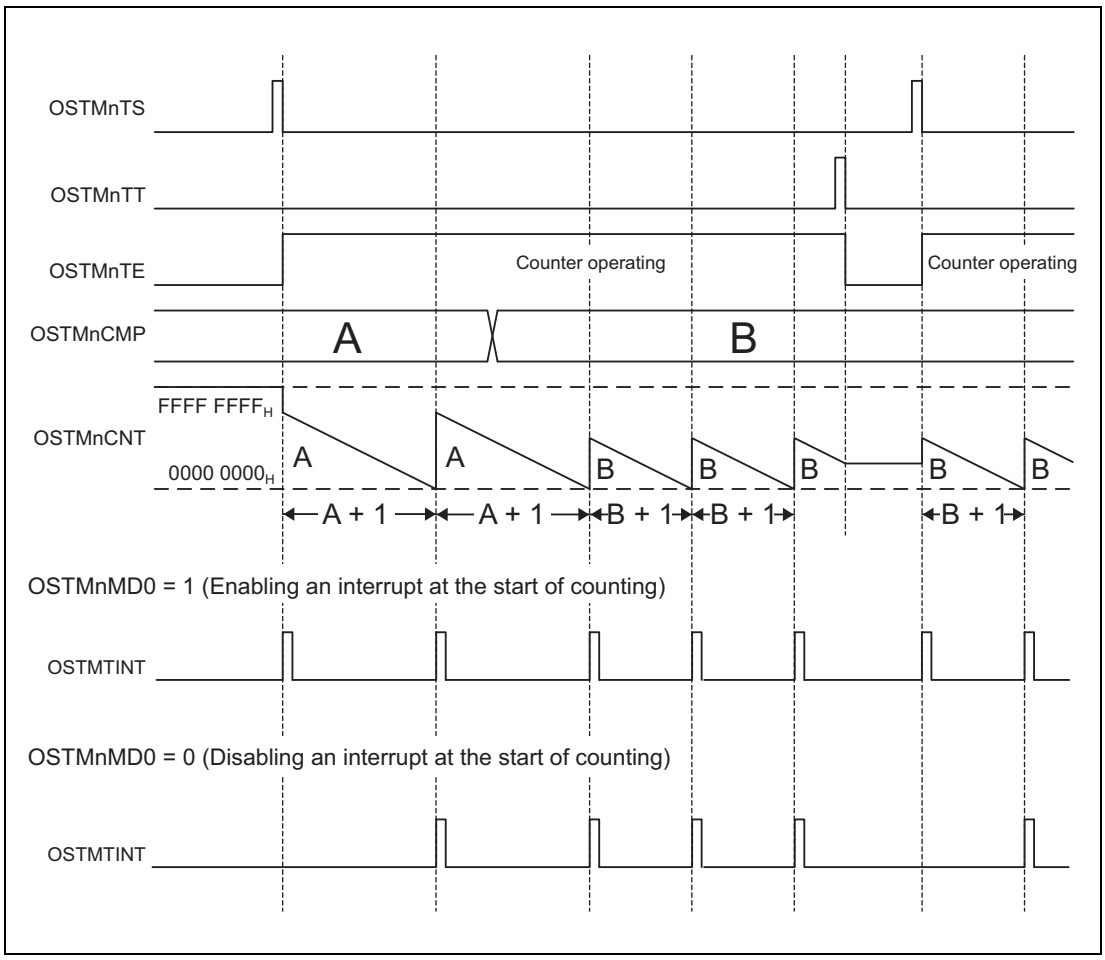



**Figure 23.2 Generating an Interrupt when Counting Starts (in Interval Timer Mode)**

## 23.3 Registers

### 23.3.1 List of Registers

OSTM registers are listed in the following table.

For details about <OSTMn_base>, see **Section 23.1.2, Register Base Address**.

**Table 23.8 Registers**

| Module | Register | Symbol | Address |
|---|---|---|---|
| OSTMn | OSTMn compare register | OSTMnCMP | <OSTMn_base> + $00_H$ |
| | OSTMn counter register | OSTMnCNT | <OSTMn_base> + $04_H$ |
| | OSTMn count enable status register | OSTMnTE | <OSTMn_base> + $10_H$ |
| | OSTMn count start trigger register | OSTMnTS | <OSTMn_base> + $14_H$ |
| | OSTMn count stop trigger register | OSTMnTT | <OSTMn_base> + $18_H$ |
| | OSTMn control register | OSTMnCTL | <OSTMn_base> + $20_H$ |
| | OSTMn emulation register | OSTMnEMU | <OSTMn_base> + $24_H$ |

### 23.3.2 OSTMnCMP — OSTMn Compare Register

This register stores the start value of the down-counter or the value with which the counter is compared, depending on the operation mode.

**Access:** This register can be read or written in 32-bit units.

**Address:** <OSTMn_base> + $00_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OSTMnCMP[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OSTMnCMP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 23.9 OSTMnCMP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | OSTMnCMP [31:0] | • In interval timer mode: start value of the down-counter<br>• In free-run compare mode: compare value |

### 23.3.3 OSTMnCNT — OSTMn Counter Register

This register indicates the counter value of the timer.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <OSTMn_base> + $04_H$

**Value after reset:** FFFF $FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OSTMnCNT[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OSTMnCNT[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 23.10 OSTMnCNT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | OSTMnCNT [31:0] | Timer counter value |

**Table 23.11** lists the correspondence among the OSTM operating mode, counting direction, and start value. The start value indicates the value to be read after the operating mode is changed.

**Table 23.11 Correspondence among Operating Mode, Counting Direction, and Start Value**

| Timer Operating Mode | OSTMnCTL.OSTMnMD1 | Counting Direction | Start Value |
|---|---|---|---|
| Interval timer mode | 0*1 | Down | FFFF $FFFF_H$ |
| Free-run compare mode | 1 | Up | 0000 $0000_H$ |

Note 1. Value after reset.

### 23.3.4 OSTMnTE — OSTMn Count Enable Status Register

This register indicates whether the counter is enabled or disabled.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <OSTMn_base> + $10_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | OSTMnTE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 23.12 OSTMnTE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | OSTMnTE | Indicates whether the counter is enabled or disabled:<br>0: Counter disabled<br>1: Counter enabled<br>This bit is set to 1 in response to OSTMnTS.OSTMnTS being set to 1.<br>Setting OSTMnTT.OSTMnTT to 1 resets this bit to 0. |

**NOTE**

If the counter is disabled, the counter value retains its value.

If the counter is restarted, it

- restarts counting down from the value in the OSTMnCMP register if it is in interval timer mode or
- restarts counting up from the counter value 0000 $0000_H$ if it is in free-run compare mode.

### 23.3.5 OSTMnTS — OSTMn Count Start Trigger Register

This register starts the counter.

**Access:** This register is a write-only register that can be written in 8-bit units.

**Address:** <OSTMn_base> + $14_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | OSTMnTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 23.13 OSTMnTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | OSTMnTS | Starts the counter:<br>0: This setting is invalid.<br>1: Starts the counter and sets OSTMnTE.OSTMnTE = 1.<br>• In interval timer mode, a forced restart is executed if this bit is set while OSTMnTE.OSTMnTE = 1.<br>• In free-run compare mode, setting this bit is ignored as long as OSTMnTE.OSTMnTE = 1. |

### 23.3.6 OSTMnTT — OSTMn Count Stop Trigger Register

This register stops the counter.

**Access:** This register is a write-only register that can be written in 8-bit units.

**Address:** <OSTMn_base> + $18_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | OSTMnTT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 23.14 OSTMnTT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | OSTMnTT | Stops the counter:<br>0: This setting is invalid.<br>1: Stops the counter and clears the OSTMnTE.OSTMnTE bit. |

### 23.3.7 OSTMnCTL — OSTMn Control Register

This register specifies the operating mode for the counter and controls the generation of OSTMTINT interrupt requests when counting starts.

Although this register is readable and writable, writing to it is only possible when OSTMnTE.OSTMnTE = 0; that is, the register becomes read only when OSTMnTE.OSTMnTE = 1.

**Access:** This register can be read or written in 8-bit units.

**Address:** <OSTMn_base> + $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | OSTMnMD1 | OSTMnMD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 23.15 OSTMnCTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | OSTMnMD1 | Specifies the operating mode for the counter:<br>0: Interval timer mode<br>1: Free-run compare mode |
| 0 | OSTMnMD0 | Controls the generation of OSTMTINT interrupt requests at the start of counting:<br>0: Interrupts when counting starts are disabled.<br>1: Interrupts when counting starts are enabled. |

### 23.3.8 OSTMnEMU — OSTMn Emulation Register

This register controls operation in combination with SVSTOP.

**Access:** This register can be read or written in 8-bit units.
Only proceed with writing while the counter is stopped (OSTMnTE.OSTMnTE = 0 and EPC.SVSTOP = 0).

**Address:** <OSTMn_base> + $24_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | OSTMnSVSDIS | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R |

**Table 23.16 OSTMnEMU Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | OSTMnSVSDIS | When EPC.SVSTOP = 0<br>Supply of the count clock is continued when the debugger acquires control of the microcontroller (at breakpoints and so on) regardless of the value of this bit (0/1).<br>When EPC.SVSTOP = 1<br>0: The count clock is stopped when the debugger acquires control of the microcontroller (at breakpoints and so on).<br>1: Supply of the count clock is continued when the debugger acquires control of the microcontroller (at breakpoints and so on). |
| 6 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

# 23.4 Operation

## 23.4.1 Starting and Stopping OSTM

OSTM is started and stopped as follows:

### Starting the timer

OSTM is started by the following setting.

- Setting the OSTMnTS.OSTMnTS bit to 1

The OSTMnTE.OSTMnTE status bit is set to 1.

The counter starts to count up or down in accordance with the settings for operating mode. For details, see **Section 23.4.2, Interval Timer Mode** and **Section 23.4.3, Free-Run Compare Mode**.

### Stopping the timer

Setting the OSTMnTT.OSTMnTT bit to 1 stops OSTM.

This also clears the OSTMnTE.OSTMnTE status bit.

## 23.4.2 Interval Timer Mode

In interval timer mode, OSTM can be used as a reference timer generating interrupt requests at fixed intervals.

### 23.4.2.1 Basic Operation in Interval Timer Mode

In interval timer mode, the timer counts down from the value specified in the OSTMnCMP register. An OSTMTINT interrupt request is generated when the counter underflows (reaches 0000 $0000_H$).

To select interval timer mode, set OSTMnCTL.OSTMnMD1 = 0.

New values can be written to the OSTMnCMP register at any time. If it is rewritten during count operation, the counter loads the new OSTMnCMP value when the next 0000 $0000_H$ is reached. Then the counter continues with the new value.

#### OSTMTINT period

The periods of OSTMTINT is:

- OSTMTINT generation period = count clock period × (OSTMnCMP + 1)

The following figure shows the basic operation of OSTM when counter-start interrupts is enabled in interval timer mode.

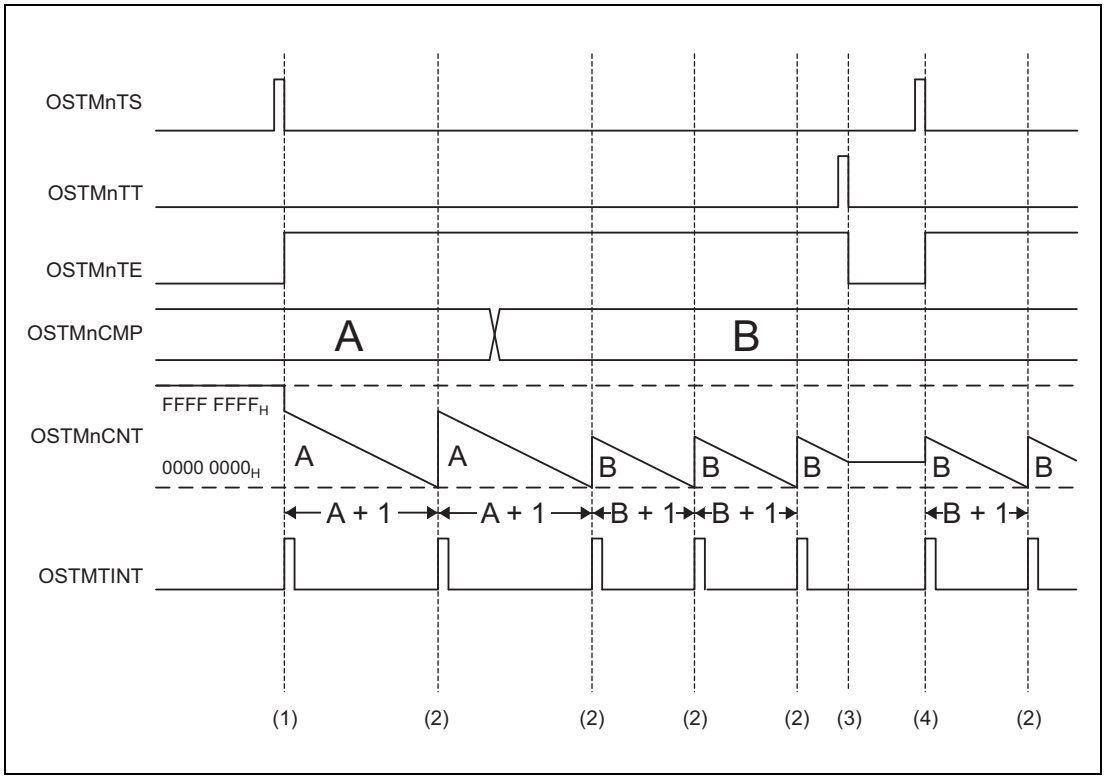



**Figure 23.3 Timing Diagram of OSTM in Interval Timer Mode**

The timing diagram above shows the following:

(1) The counter starts counting when OSTMnTS.OSTMnTS = 1. The OSTMnTE.OSTMnTE bit is set to indicate enabling of the counter.
The counter starts counting down from the value of OSTMnCMP.
If OSTMnCTL.OSTMnMD0 is 1, OSTMTINT interrupt requests are generated at the start of counting. The OSTMnCNT register indicates the counter value.

(2) When the counter reaches 0000 0000$_H$, an OSTMTINT interrupt request is generated. The counter loads the new start value from OSTMnCMP and continues counting down.

(3) When the counter is stopped (OSTMnTT.OSTMnTT = 1), the OSTMnTE.OSTMnTE bit is cleared to indicate disabling of the counter. The counter retains its current value until it is restarted.

(4) When counting is restarted (OSTMnTS.OSTMnTS = 1), the counter loads the new start value from OSTMnCMP and starts counting down.

### Forced restart

The counter is forcibly restarted by setting OSTMnTS.OSTMnTS = 1 during counting.

The counter loads the start value from the OSTMnCMP register and continues to count down.

The following figure shows the timing in interval timer mode, with counter-start interrupts enabled (OSTMnCTL.OSTMnMD0 = 1).

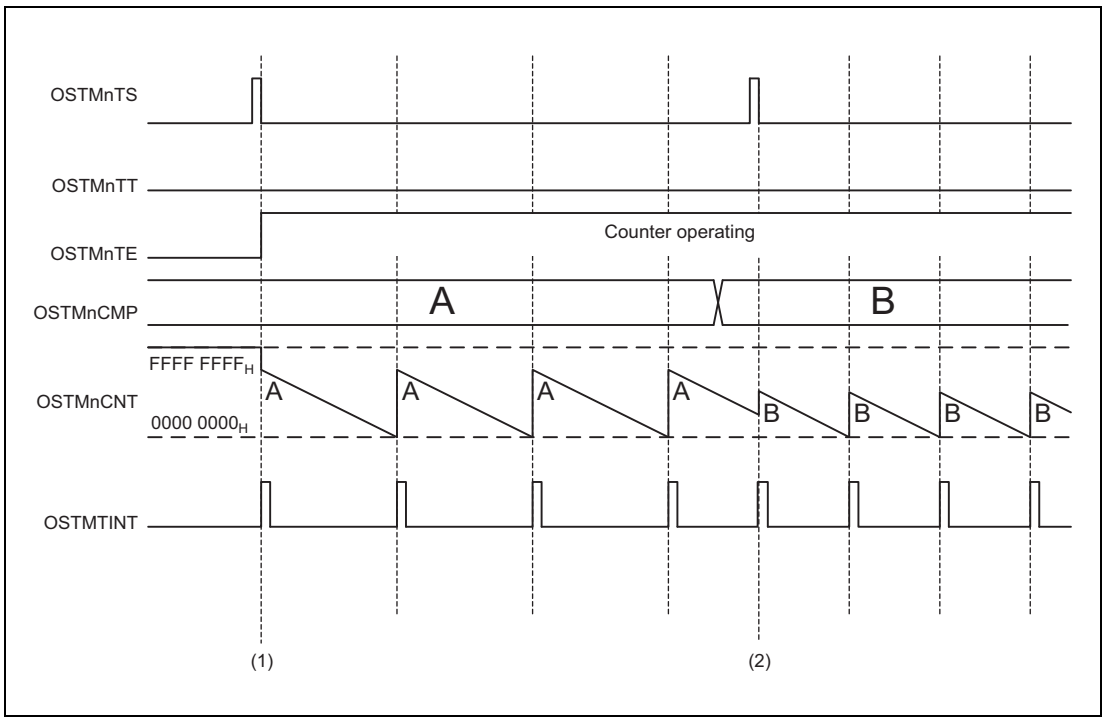



**Figure 23.4 Timing Diagram of Forced Restart in Interval Timer Mode**

The timing diagram above shows the following operations:

(1) The counter is started and stopped as described under **Figure 23.3, Timing Diagram of OSTM in Interval Timer Mode**.

(2) Setting OSTMnTS.OSTMnTS = 1 restarts the counter while counting is in progress (i.e. while OSTMnTE.OSTMnTE = 1).
The counter immediately restarts counting down, starting with the current value of OSTMnCMP.
When OSTMnCTL.OSTMnMD0 = 1, an OSTMTINT interrupt request is generated when counting starts.

### 23.4.2.2 Operation when OSTMnCMP = 0000 0000$_H$

When OSTMnCMP = 0000 0000$_H$, OSTM behaves as follows.

- When the counter is enabled, the OSTMTINT interrupt request is always set to 1.

The following figure shows operations of OSTM when OSTMnCMP = 0000 0000$_H$, and counter-start interrupts are enabled.

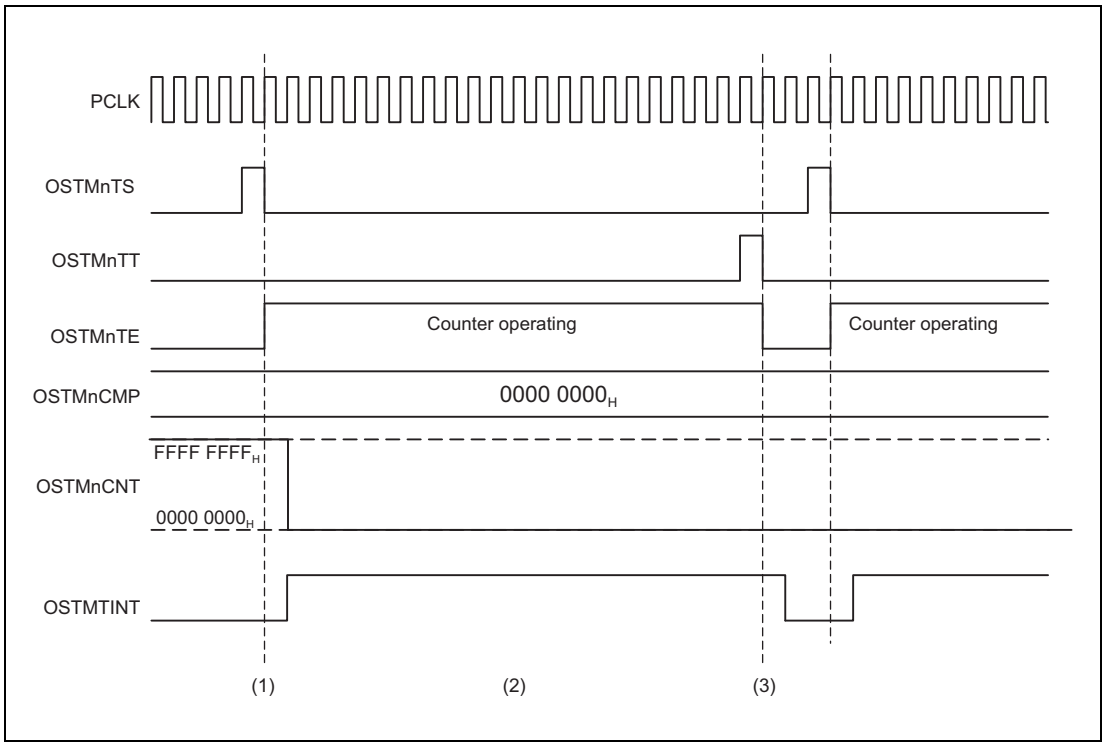



**Figure 23.5 Timing Diagram when OSTMnCMP = 0000 0000$_H$ in Interval Timer Mode**

The timing diagram above shows the following operations:

(1) The counter is reloaded with the value in OSTMnCMP as soon as it starts counting, so the value 0000 0000$_H$ is retained in OSTMnCMP.

(2) The OSTMTINT interrupt request is continuously asserted.

(3) After the counter stops, the OSTMTINT interrupt request signal is deasserted.

When interrupts on starting of the counter are disabled, no interrupt is generated when counting starts.

### 23.4.2.3 Setting Procedure for Interval Timer Mode

The setting procedure in interval timer mode after reset release is described below:

#### Setting procedure

(1) Set the start value of the counter in the OSTMnCMP register.

(2) Select interval timer mode by setting the OSTMnCTL.OSTMnMD1 bit to 0.

(3) Enable or disable interrupts when counting starts (OSTMnCTL.OSTMnMD0).

## 23.4.3 Free-Run Compare Mode

### 23.4.3.1 Basic Operation in Free-Run Compare Mode

In free-run compare mode, the counter counts up from 0000 $0000_H$ to FFFF $FFFF_H$. When the value of the OSTMnCMP register matches the current counter value, an OSTMTINT interrupt request is output.

In free-run compare mode, set OSTMnCTL.OSTMnMD1 = 1.

New values can be written to the OSTMnCMP register at any time.

The following figure shows the basic operation of OSTM in free-run compare mode with the start of counting enabled (OSTMnCTL.OSTMnMD0 = 1).

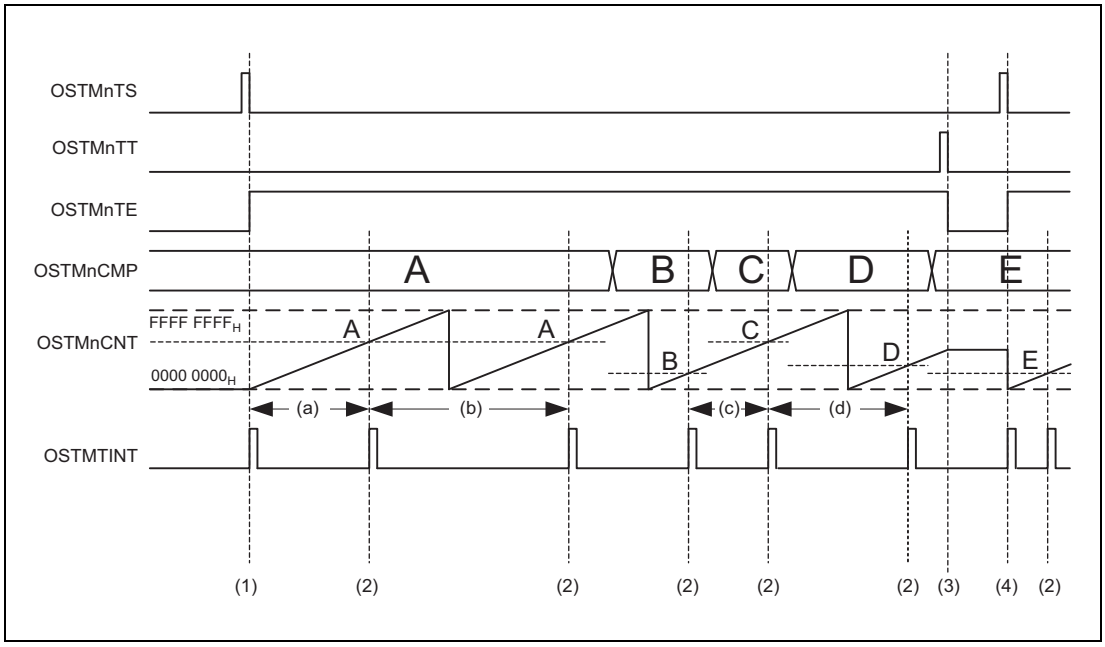



**Figure 23.6 Timing Diagram of OSTM in Free-Run Compare Mode**

The timing diagram above shows the following:

(1) The counter starts counting when OSTMnTS.OSTMnTS = 1.
The OSTMnTE.OSTMnTE bit is set to indicate enabling of the counter. The counter counts up from 0000 $0000_H$ to FFFF $FFFF_H$. The OSTMnCNT register indicates the counter value.
When OSTMnCTL.OSTMnMD0 = 1, an OSTMTINT interrupt request is generated at the start of counting.

(2) When the current counter value matches the value in the OSTMnCMP register, an OSTMTINT interrupt request is generated.

(3) When the counter is stopped (OSTMnTT.OSTMnTT = 1), the OSTMnTE.OSTMnTE bit is cleared to indicate disabling of the counter.
The counter retains its current value until it is restarted.

(4) Counting by the counter restarts from 0000 $0000_H$ when OSTMnTS.OSTMnTS = 1.

### OSTMTINT period

The OSTMTINT generation period is different depending on the starting time. If OSTMnCMP is rewritten during operation, the period is changed according to the size of the new and old compare values.

**Table 23.17 OSTMTINT Generation Timing**

| Old Value for Comparison | New Value for Comparison | Counter Value at Time of Rewriting | Period of OSTMTINT Generation | Label in Timing Diagram |
|---|---|---|---|---|
| Counter starts | | | (A + 1) × count clock period | (a) |
| A | A | No rewriting | (FFFF FFFF$_H$ + 1) × count clock period | (b) |
| B | C > B | B < counter value < C | (C – B) × count clock period | (c) |
| C | D < C | Counter value > D,C | (FFFF FFFF$_H$ – C + D + 1) × count clock period | (d) |

### Forced restart

Forced restarting of the counter does not proceed if the OSTMnTS.OSTMnTS bit is set during counting. The counter ignores the attempted setting and continues counting.

## 23.4.3.2 Operation when OSTMnCMP = 0000 0000$_H$

The following figure shows the operation of OSTM when OSTMnCMP = 0000 0000$_H$, and counter-start interrupts are enabled (OSTMnCTL.OSTMnMD0 = 1).

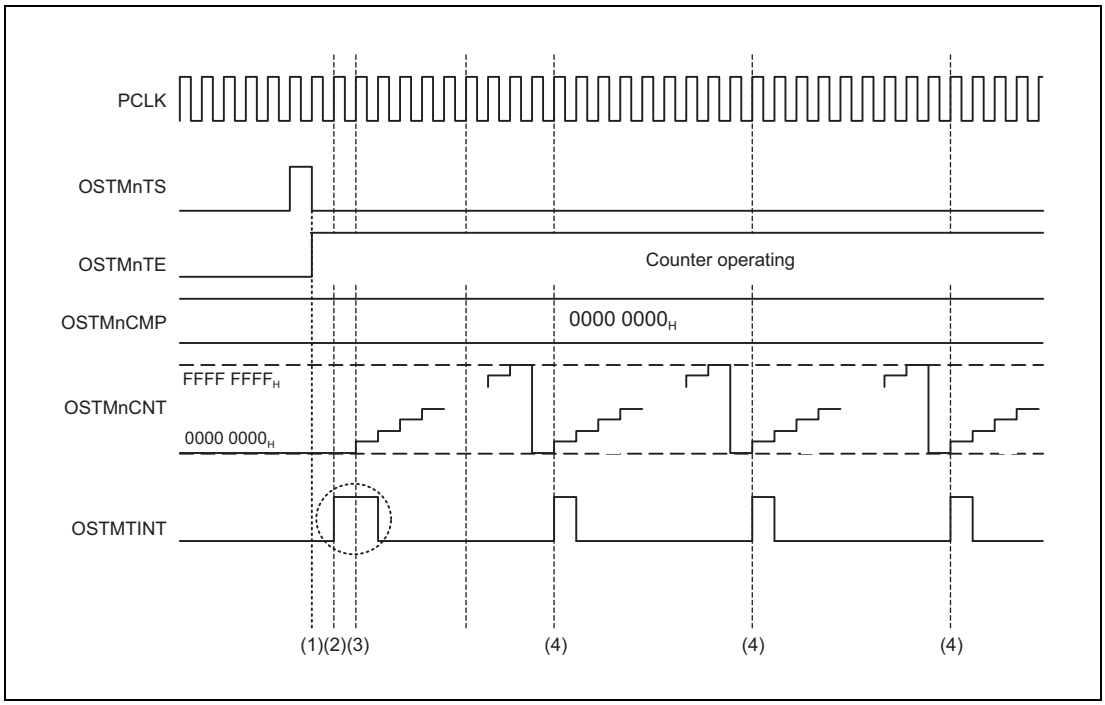



**Figure 23.7 Timing Diagram when OSTMnCMP = 0000 0000$_H$ in Free-Run Compare Mode**

The timing diagram above shows the following operations.

(1) Once the counter starts, it counts up from 0000 0000$_H$ to FFFF FFFF$_H$.

(2) An OSTMTINT interrupt request is generated when counting starts.

(3) If the current counter value matches OSTMnCMP, an OSTMTINT interrupt request is generated. If OSTMnCMP = 0000 $0000_H$ as shown above, OSTMTINT is generated over two clock cycles.

(4) An OSTMTINT interrupt request is generated for each clock cycle (FFFF $FFFF_H$ +1).

When interrupts on starting of the counter are disabled, no interrupt is generated when counting starts.

#### 23.4.3.3 Setting Procedure for Free-Run Compare Mode

The setting procedure in free-run compare mode after reset release is described below:

**Setting procedure**

(1) Set the compare value in the OSTMnCMP register.

(2) Select free-run compare mode by setting the OSTMnCTL.OSTMnMD1 bit to 1.

(3) Enable or disable interrupts when counting starts by the OSTMnCTL.OSTMnMD0 bit.

# Section 24 Timer Array Unit B (TAUB)

This section contains a generic description of the timer array unit B (TAUB).

The first part in this section describes the RH850/F1K specific features, such as the number of units and the register base addresses. The remainder of the section describes the functions and registers of the TAUB.

## 24.1 Features of RH850/F1K TAUB

### 24.1.1 Number of Units and Channels

This microcontroller has the following number of TAUB units.

**Table 24.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of Units | 1 | 1 | 2 |
| Name | TAUBn (n = 0) | TAUBn (n = 0) | TAUBn (n = 0, 1) |

TAUBn has the following number of channels of timers.

**Table 24.2 TAUBn Unit Configurations and Channels**

| Unit Name (Channel Name) TAUBn | Channels per Unit | RH850/F1K 100 pins (16 ch) | RH850/F1K 144 pins (16 ch) | RH850/F1K 176 pins (32 ch) |
|---|---|---|---|---|
| TAUB0 | 16 | √ | √ | √ |
| TAUB1 | 16 | - | - | √ |

**Table 24.3 Indices**

| Index | Description |
|---|---|
| n | Throughout this section, the individual TAUB units are identified by the index “n”; for example, TAUBnTOM is the TAUBn channel output mode register. |
| m | The TAUB has 16 channels. Throughout this section, the individual channels are identified by the index “m” (m = 0 to 15), thus a certain channel is denoted as CHm.<br>The even numbered channels (m = 0, 2, 4, 6, 8, 10, 12, 14) are denoted as CHm_even.<br>The odd numbered channels (m = 1, 3, 5, 7, 9, 11, 13, 15) are denoted as CHm_odd. |

### 24.1.2 Register Base Addresses

TAUBn base addresses are listed in the following table.

TAUBn register addresses are given as offsets from the base addresses.

**Table 24.4 Register Base Addresses**

| Name | Base Address |
|---|---|
| <TAUB0_base> | FFE3 $0000_H$ |
| <TAUB1_base> | FFE3 $1000_H$ |

### 24.1.3 Clock Supply

The TAUBn clock supply is shown in the following table.

**Table 24.5 TAUBn Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| TAUBn | PCLK | CKSCLK_IPERI2 | Timer count clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CKSCLK_IPERI2 | |

### 24.1.4 Interrupt Requests

TAUBn interrupt requests are listed in the following table.

**Table 24.6 Interrupt Requests (1/2)**

| Unit Interrupt Name | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **TAUB0** | | | |
| INTTAUB0I0 | Channel 0 interrupt | 142 | 33 |
| INTTAUB0I1 | Channel 1 interrupt | 143 | 92 |
| INTTAUB0I2 | Channel 2 interrupt | 144 | 34 |
| INTTAUB0I3 | Channel 3 interrupt | 145 | 93 |
| INTTAUB0I4 | Channel 4 interrupt | 146 | 35 |
| INTTAUB0I5 | Channel 5 interrupt | 147 | 94 |
| INTTAUB0I6 | Channel 6 interrupt | 148 | 36 |
| INTTAUB0I7 | Channel 7 interrupt | 149 | 95 |
| INTTAUB0I8 | Channel 8 interrupt | 150 | 96 |
| INTTAUB0I9 | Channel 9 interrupt | 151 | 37 |
| INTTAUB0I10 | Channel 10 interrupt | 152 | 97 |
| INTTAUB0I11 | Channel 11 interrupt | 153 | 38 |
| INTTAUB0I12 | Channel 12 interrupt | 154 | 98 |
| INTTAUB0I13 | Channel 13 interrupt | 155 | 39 |
| INTTAUB0I14 | Channel 14 interrupt | 156 | 99 |
| INTTAUB0I15 | Channel 15 interrupt | 157 | 40 |
| **TAUB1** | | | |
| INTTAUB1I0 | Channel 0 interrupt | 256 | 52 |
| INTTAUB1I1 | Channel 1 interrupt | 257 | 115 |
| INTTAUB1I2 | Channel 2 interrupt | 258 | 53 |
| INTTAUB1I3 | Channel 3 interrupt | 259 | 116 |
| INTTAUB1I4 | Channel 4 interrupt | 260 | 54 |
| INTTAUB1I5 | Channel 5 interrupt | 261 | 117 |
| INTTAUB1I6 | Channel 6 interrupt | 262 | 55 |
| INTTAUB1I7 | Channel 7 interrupt | 263 | 118 |
| INTTAUB1I8 | Channel 8 interrupt | 264 | 119 |
| INTTAUB1I9 | Channel 9 interrupt | 265 | 56 |
| INTTAUB1I10 | Channel 10 interrupt | 266 | 120 |
| INTTAUB1I11 | Channel 11 interrupt | 267 | 57 |

Table 24.6 Interrupt Requests (2/2)

| Unit Interrupt Name | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| INTTAUB1I12 | Channel 12 interrupt | 268 | 121 |
| INTTAUB1I13 | Channel 13 interrupt | 269 | 58 |
| INTTAUB1I14 | Channel 14 interrupt | 270 | 122 |
| INTTAUB1I15 | Channel 15 interrupt | 271 | 59 |

### 24.1.5 Reset Sources

TAUBn reset sources are listed in the following table. TAUBn is initialized by these reset sources.

Table 24.7 Reset Sources

| Unit Name | Reset Source |
|---|---|
| TAUBn | All reset sources (ISORES) |

### 24.1.6 External input/output Signals

External input/output signals of TAUBn are listed below.

Table 24.8 External Input/Output Signals (1/2)

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **TAUB0** | | |
| TAUBTTIN0 | Channel 0 input*1 | TAUB0I0 |
| TAUBTTIN1 | Channel 1 input*1 | TAUB0I1 |
| TAUBTTIN2 | Channel 2 input*1 | TAUB0I2 |
| TAUBTTIN3 | Channel 3 input*1 | TAUB0I3 |
| TAUBTTIN4 | Channel 4 input*1 | TAUB0I4 |
| TAUBTTIN5 | Channel 5 input*1 | TAUB0I5 |
| TAUBTTIN6 | Channel 6 input*1 | TAUB0I6 |
| TAUBTTIN7 | Channel 7 input*1 | TAUB0I7 |
| TAUBTTIN8 | Channel 8 input*1 | TAUB0I8 |
| TAUBTTIN9 | Channel 9 input*1 | TAUB0I9 |
| TAUBTTIN10 | Channel 10 input*1 | TAUB0I10 |
| TAUBTTIN11 | Channel 11 input*1 | TAUB0I11 |
| TAUBTTIN12 | Channel 12 input*1 | TAUB0I12 |
| TAUBTTIN13 | Channel 13 input*1 | TAUB0I13 |
| TAUBTTIN14 | Channel 14 input*1 | TAUB0I14 |
| TAUBTTIN15 | Channel 15 input*1 | TAUB0I15 |
| TAUBTTOUT0 | Channel 0 output | TAUB0O0 |
| TAUBTTOUT1 | Channel 1 output | TAUB0O1 |
| TAUBTTOUT2 | Channel 2 output | TAUB0O2 |
| TAUBTTOUT3 | Channel 3 output | TAUB0O3 |
| TAUBTTOUT4 | Channel 4 output | TAUB0O4 |
| TAUBTTOUT5 | Channel 5 output | TAUB0O5 |

Table 24.8 External Input/Output Signals (2/2)

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| TAUBTTOUT6 | Channel 6 output | TAUB0O6 |
| TAUBTTOUT7 | Channel 7 output | TAUB0O7 |
| TAUBTTOUT8 | Channel 8 output | TAUB0O8 |
| TAUBTTOUT9 | Channel 9 output | TAUB0O9 |
| TAUBTTOUT10 | Channel 10 output | TAUB0O10 |
| TAUBTTOUT11 | Channel 11 output | TAUB0O11 |
| TAUBTTOUT12 | Channel 12 output | TAUB0O12 |
| TAUBTTOUT13 | Channel 13 output | TAUB0O13 |
| TAUBTTOUT14 | Channel 14 output | TAUB0O14 |
| TAUBTTOUT15 | Channel 15 output | TAUB0O15 |
| **TAUB1** | | |
| TAUBTTIN0 | Channel 0 input*1 | TAUB1I0 |
| TAUBTTIN1 | Channel 1 input*1 | TAUB1I1 |
| TAUBTTIN2 | Channel 2 input*1 | TAUB1I2 |
| TAUBTTIN3 | Channel 3 input*1 | TAUB1I3 |
| TAUBTTIN4 | Channel 4 input*1 | TAUB1I4 |
| TAUBTTIN5 | Channel 5 input*1 | TAUB1I5 |
| TAUBTTIN6 | Channel 6 input*1 | TAUB1I6 |
| TAUBTTIN7 | Channel 7 input*1 | TAUB1I7 |
| TAUBTTIN8 | Channel 8 input*1 | TAUB1I8 |
| TAUBTTIN9 | Channel 9 input*1 | TAUB1I9 |
| TAUBTTIN10 | Channel 10 input*1 | TAUB1I10 |
| TAUBTTIN11 | Channel 11 input*1 | TAUB1I11 |
| TAUBTTIN12 | Channel 12 input*1 | TAUB1I12 |
| TAUBTTIN13 | Channel 13 input*1 | TAUB1I13 |
| TAUBTTIN14 | Channel 14 input*1 | TAUB1I14 |
| TAUBTTIN15 | Channel 15 input*1 | TAUB1I15 |
| TAUBTTOUT0 | Channel 0 output | TAUB1O0 |
| TAUBTTOUT1 | Channel 1 output | TAUB1O1 |
| TAUBTTOUT2 | Channel 2 output | TAUB1O2 |
| TAUBTTOUT3 | Channel 3 output | TAUB1O3 |
| TAUBTTOUT4 | Channel 4 output | TAUB1O4 |
| TAUBTTOUT5 | Channel 5 output | TAUB1O5 |
| TAUBTTOUT6 | Channel 6 output | TAUB1O6 |
| TAUBTTOUT7 | Channel 7 output | TAUB1O7 |
| TAUBTTOUT8 | Channel 8 output | TAUB1O8 |
| TAUBTTOUT9 | Channel 9 output | TAUB1O9 |
| TAUBTTOUT10 | Channel 10 output | TAUB1O10 |
| TAUBTTOUT11 | Channel 11 output | TAUB1O11 |
| TAUBTTOUT12 | Channel 12 output | TAUB1O12 |
| TAUBTTOUT13 | Channel 13 output | TAUB1O13 |
| TAUBTTOUT14 | Channel 14 output | TAUB1O14 |
| TAUBTTOUT15 | Channel 15 output | TAUB1O15 |

Note 1. When channel input pins are to be used, noise filters must be set for the corresponding port pin functions. For details, see **Section 2.12, Noise Filter & Edge/Level Detector**.

## 24.2 Overview

### 24.2.1 Functional Overview

The TAUB has the following functions:

- Independent channel operation function (operated using a single channel)
- Synchronous channel operation function (operated using a master channel and multiple slave channels)

The TAUB is used to perform various count or timer operations and to output a signal which depends on the result of the operation. It contains one prescaler block for count clock generation and 16 channels, each equipped with a 16-bit counter TAUBnCNTm and a 16-bit data register TAUBnCDRm to hold the count start value and compare value.

It also contains several control and status registers.

#### Independent and synchronous operation

Every channel can operate either independently or in combination with other channels (synchronously). When one master channel and one or more slave channels operate in combination, the slave channels depend on the master channel.

When a channel is operated independently, it can be operated independent of all other channels. The synchronous operation function is implemented using a combination of channel groups (consisted of master and slave channels).

Several rules apply to the settings of channels.

## 24.2.2 Terms

In this section, the following terms are used:

### Independent channel operation function/synchronous channel operation function

TAUB has 16 channels, and provides an independent channel operation function whereby individual channels operate independently and a synchronous channel operation function whereby multiple channels operate in combination.

- The independent channel operation function can be used by any channel independently of all other channels.
- The synchronous channel operation function is implemented using a combination of channel groups (comprised of master and slave channels).

Several rules apply to the settings of channels.

### Channel group

In the synchronous channel operation function, all channels that depend on each other are referred to as a “channel group”.
A channel group has one master channel and one or more slave channels.

### Upper/lower channel

Depending on the channel number m, a channel with a smaller number or with a larger number is referred to as “upper” or “lower” channel, respectively.

- Upper channel: Channel with a smaller channel number
- Lower channel: Channel with a larger channel number

Example:
For channel 5, channel 3 is an upper channel and channel 9 is a lower channel. Channel 0 is the highest channel and channel 15 is the lowest channel.

### 24.2.3 Functional List of Timer Operations

This timer provides the following functions by operating each channel independently or by combining multiple channels.

**Table 24.9 Functional List of TAUB Operations**

| Operation Function | | Setting Example |
|---|---|---|
| **Independent Channel Operation Functions** | | **Section 24.12** |
| | Interval Timer Function | **Section 24.12.1** |
| | TAUBTTINm Input Interval Timer Function | **Section 24.12.2** |
| | Clock Divide Function | **Section 24.12.3** |
| | External Event Count Function | **Section 24.12.4** |
| | One-Pulse Output Function | **Section 24.12.5** |
| | TAUBTTINm Input Pulse Interval Measurement Function | **Section 24.12.6** |
| | TAUBTTINm Input Signal Width Measurement Function | **Section 24.12.7** |
| | TAUBTTINm Input Position Detection Function | **Section 24.12.8** |
| | TAUBTTINm Input Period Count Detection Function | **Section 24.12.9** |
| | TAUBTTINm Input Pulse Interval Judgment Function | **Section 24.12.10** |
| | TAUBTTINm Input Signal Width Judgment Function | **Section 24.12.11** |
| | Overflow Interrupt Output Function (during TAUBTTINm Width Measurement) | **Section 24.12.12** |
| | Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection) | **Section 24.12.13** |
| **Independent Channel Simultaneous Rewrite Functions** | | **Section 24.13** |
| | Simultaneous Rewrite Trigger Generation Function Type 1 | **Section 24.13.1** |
| **Synchronous Channel Operation Functions** | | **Section 24.14** |
| | PWM Output Function | **Section 24.14.1** |
| | One-Shot Pulse Output Function | **Section 24.14.2** |
| | Delay Pulse Output Function | **Section 24.14.3** |
| | AD Conversion Trigger Output Function Type 1 | **Section 24.14.4** |
| | Triangle PWM Output Function | **Section 24.14.5** |
| | Triangle PWM Output Function with Dead Time | **Section 24.14.6** |
| | A/D Conversion Trigger Output Function Type 2 | **Section 24.14.7** |

### 24.2.4 Input/Output Interrupt Request Signals

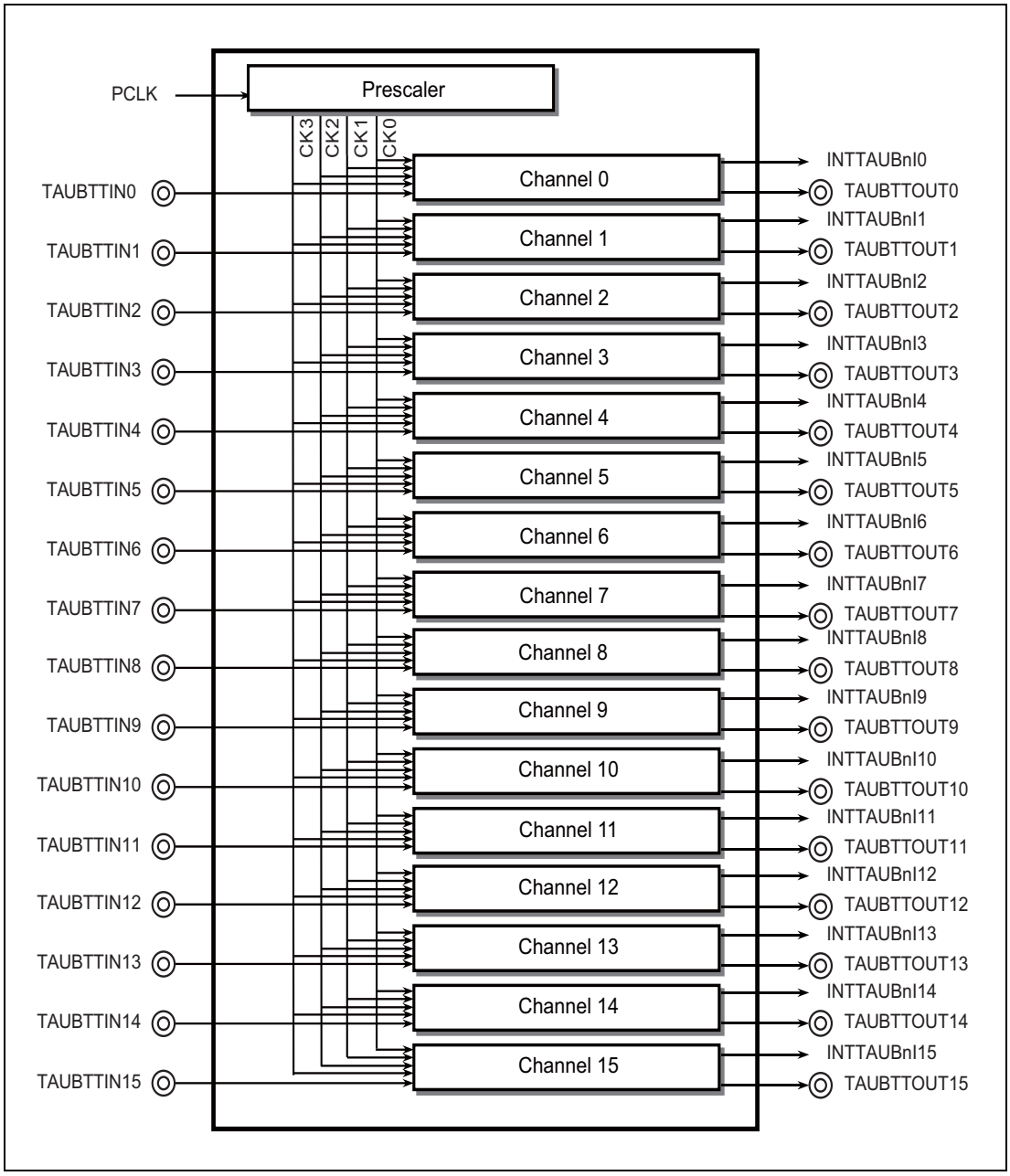



**Figure 24.1 TAUB Input/Output and Interrupt Request Signals**

### 24.2.5 Block Diagram

The following figure shows the main components of the TAUB.

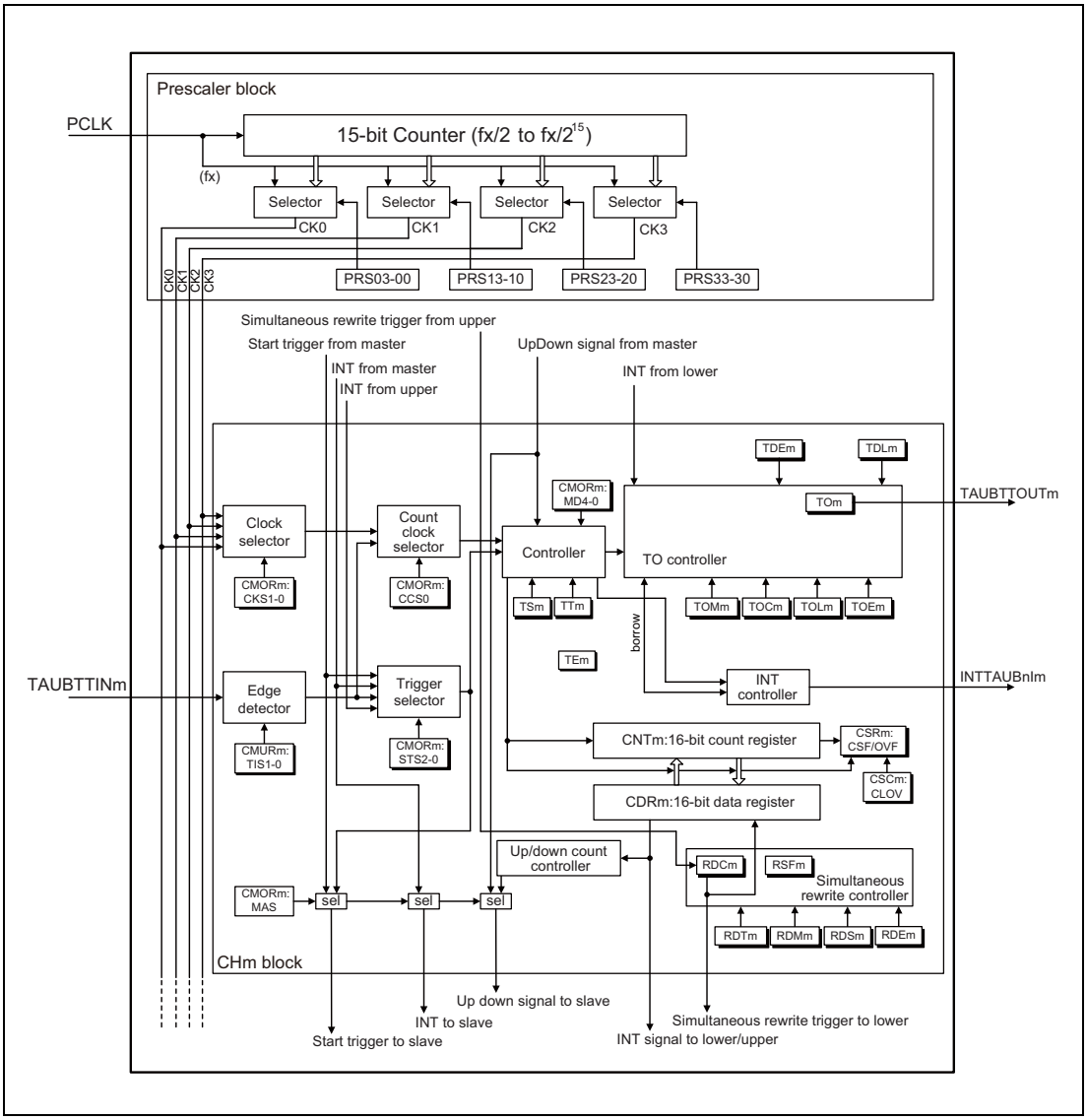



**Figure 24.2 Block Diagram of the TAUB**

The prefix “TAUBn” has been omitted from the register names for the sake of clarity in the above figure.

## 24.2.6 Description of Blocks

The following describes the functional blocks:

### Prescaler block

The prescaler block provides up to 4 clock signals (CK0 to CK3) that can be used as count clocks for all channels.

Count clocks CK0 to CK3 are derived from PCLK by a configurable prescaler division factor of $2^0$ to $2^{15}$.

### Clock and count clock selection

For every channel, the count clock selector selects which of the following is used as the clock source:

- One of the clocks CK0 to CK3 (selected by the clock selector)
- TAUBTTINm input signal valid edge

### Controller

The controller controls the main operations of the counter:

- Operation mode (selected by bits TAUBnCMORm.TAUBnMD[4:0])
- Counter start enable (TAUBnTS.TAUBnTSm) and counter stop (TAUBnTT.TAUBnTTm)
  When counter start is enabled, status flag TAUBnTE.TAUBnTEm is set.
- Count direction (up/down) (can be controlled by the master channel)

### Trigger selector

Depending on the selected operation mode, the counter starts automatically when it is enabled (TAUBnTE.TAUBnTEm = 1), or it waits for an external start trigger signal. Any of the following signals can be used as the start trigger:

- Input of TAUBTTINm valid edge
- INTTAUBnIm from the master or any upper channel
- Up/down output trigger signal of the master channel
- Dead-time output signal of the TAUBTTOUTm generation unit

### Simultaneous rewrite controller

Simultaneous rewrite control is a special function that can be used if multiple cahnnels like synchronous operation functions are used. The data registers (TAUBnCDRm) of all channels in a channel group can be rewritten at any time. The simultaneous rewrite controller ensures that new data register values of all channels become effective at the same time.

### TAUBnTO controller

The output control of every channel enables the generation of various output signal forms such as PWM signals or triangular waves.

# 24.3 Registers

## 24.3.1 List of Registers

TAUB registers are listed in the following table.

For details about <TAUBn_base>, see **Section 24.1.2, Register Base Addresses**.

**Table 24.10 List of Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| **TAUBn prescaler registers** | | | |
| TAUBn | TAUBn prescaler clock select register | TAUBnTPS | <TAUBn_base> + $240_H$ |
| **TAUBn control registers** | | | |
| TAUBn | TAUBn channel data register m | TAUBnCDRm | <TAUBn_base> + $0_H$ + m × $4_H$ |
| | TAUBn channel counter register m | TAUBnCNTm | <TAUBn_base> + $80_H$ + m × $4_H$ |
| | TAUBn channel mode OS register m | TAUBnCMORm | <TAUBn_base> + $200_H$ + m × $4_H$ |
| | TAUBn channel mode user register m | TAUBnCMURm | <TAUBn_base> + $C0_H$ + m × $4_H$ |
| | TAUBn channel status register m | TAUBnCSRm | <TAUBn_base> + $140_H$ + m × $4_H$ |
| | TAUBn channel status clear trigger register m | TAUBnCSCm | <TAUBn_base> + $180_H$ + m × $4_H$ |
| | TAUBn channel start trigger register | TAUBnTS | <TAUBn_base> + $1C4_H$ |
| | TAUBn channel enable status register | TAUBnTE | <TAUBn_base> + $1C0_H$ |
| | TAUBn channel stop trigger register | TAUBnTT | <TAUBn_base> + $1C8_H$ |
| **TAUBn output registers** | | | |
| TAUBn | TAUBn channel output enable register | TAUBnTOE | <TAUBn_base> + $5C_H$ |
| | TAUBn channel output register | TAUBnTO | <TAUBn_base> + $58_H$ |
| | TAUBn channel output mode register | TAUBnTOM | <TAUBn_base> + $248_H$ |
| | TAUBn channel output configuration register | TAUBnTOC | <TAUBn_base> + $24C_H$ |
| | TAUBn channel output active level register | TAUBnTOL | <TAUBn_base> + $040_H$ |
| | TAUBn channel dead time output enable register | TAUBnTDE | <TAUBn_base> + $250_H$ |
| | TAUBn channel dead time output level register | TAUBnTDL | <TAUBn_base> + $54_H$ |
| **TAUBn reload data registers** | | | |
| TAUBn | TAUBn channel reload data enable register | TAUBnRDE | <TAUBn_base> + $260_H$ |
| | TAUBn channel reload data mode register | TAUBnRDM | <TAUBn_base> + $264_H$ |
| | TAUBn channel reload data control CH select register | TAUBnRDS | <TAUBn_base> + $268_H$ |
| | TAUBn channel reload data control register | TAUBnRDC | <TAUBn_base> + $26C_H$ |
| | TAUBn channel reload data trigger register | TAUBnRDT | <TAUBn_base> + $44_H$ |
| | TAUBn channel reload status register | TAUBnRSF | <TAUBn_base> + $48_H$ |
| **TAUBn emulation register** | | | |
| TAUBn | TAUBn emulation register | TAUBnEMU | <TAUBn_base> + $290_H$ |

## 24.3.2 Details of TAUBn Prescaler Registers

### 24.3.2.1 TAUBnTPS — TAUBn Prescaler Clock Select Register

This register specifies the clocks CK0, CK1, CK2, and CK3 for all channels of the PCLK prescaler.

**Access:** This register can be read or written in 16-bit units.

**Address:** <TAUBn_base> + $240_H$

**Value after reset:** $FFFF_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnPRS3[3:0] | | | | TAUBnPRS2[3:0] | | | | TAUBnPRS1[3:0] | | | | TAUBnPRS0[3:0] | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.11 TAUBnTPS Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 12 | TAUBnPRS3 [3:0] | Specifies the CK3 clock. |

| TAUBnPRS3[3:0] | CK3 clock |
|---|---|
| $0000_B$ | PCLK/$2^0$ |
| $0001_B$ | PCLK/$2^1$ |
| $0010_B$ | PCLK/$2^2$ |
| $0011_B$ | PCLK/$2^3$ |
| $0100_B$ | PCLK/$2^4$ |
| $0101_B$ | PCLK/$2^5$ |
| $0110_B$ | PCLK/$2^6$ |
| $0111_B$ | PCLK/$2^7$ |
| $1000_B$ | PCLK/$2^8$ |
| $1001_B$ | PCLK/$2^9$ |
| $1010_B$ | PCLK/$2^{10}$ |
| $1011_B$ | PCLK/$2^{11}$ |
| $1100_B$ | PCLK/$2^{12}$ |
| $1101_B$ | PCLK/$2^{13}$ |
| $1110_B$ | PCLK/$2^{14}$ |
| $1111_B$ | PCLK/$2^{15}$ |

These bits can only be rewritten when all counters using CK3 are stopped (TAUBnTE.TAUBnTEm = 0).

**Table 24.11 TAUBnTPS Register Contents (2/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 to 8 | TAUBnPRS2 [3:0] | Specifies the CK2 clock. |

| TAUBnPRS2[3:0] | CK2 Clock |
|---|---|
| $0000_B$ | $PCLK/2^0$ |
| $0001_B$ | $PCLK/2^1$ |
| $0010_B$ | $PCLK/2^2$ |
| $0011_B$ | $PCLK/2^3$ |
| $0100_B$ | $PCLK/2^4$ |
| $0101_B$ | $PCLK/2^5$ |
| $0110_B$ | $PCLK/2^6$ |
| $0111_B$ | $PCLK/2^7$ |
| $1000_B$ | $PCLK/2^8$ |
| $1001_B$ | $PCLK/2^9$ |
| $1010_B$ | $PCLK/2^{10}$ |
| $1011_B$ | $PCLK/2^{11}$ |
| $1100_B$ | $PCLK/2^{12}$ |
| $1101_B$ | $PCLK/2^{13}$ |
| $1110_B$ | $PCLK/2^{14}$ |
| $1111_B$ | $PCLK/2^{15}$ |

These bits can only be rewritten when all counters using CK2 are stopped (TAUBnTE.TAUBnTEm = 0).

| | | |
|---|---|---|
| 7 to 4 | TAUBnPRS1 [3:0] | Specifies the CK1 clock. |

| TAUBnPRS1[3:0] | CK1 Clock |
|---|---|
| $0000_B$ | $PCLK/2^0$ |
| $0001_B$ | $PCLK/2^1$ |
| $0010_B$ | $PCLK/2^2$ |
| $0011_B$ | $PCLK/2^3$ |
| $0100_B$ | $PCLK/2^4$ |
| $0101_B$ | $PCLK/2^5$ |
| $0110_B$ | $PCLK/2^6$ |
| $0111_B$ | $PCLK/2^7$ |
| $1000_B$ | $PCLK/2^8$ |
| $1001_B$ | $PCLK/2^9$ |
| $1010_B$ | $PCLK/2^{10}$ |
| $1011_B$ | $PCLK/2^{11}$ |
| $1100_B$ | $PCLK/2^{12}$ |
| $1101_B$ | $PCLK/2^{13}$ |
| $1110_B$ | $PCLK/2^{14}$ |
| $1111_B$ | $PCLK/2^{15}$ |

These bits can only be rewritten when all counters using CK1 are stopped (TAUBnTE.TAUBnTEm = 0).

**Table 24.11 TAUBnTPS Register Contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 3 to 0 | TAUBnPRS0 [3:0] | Specifies the CK0 clock. |

| TAUBnPRS0[3:0] | CK0 Clock |
|---|---|
| $0000_B$ | $PCLK/2^0$ |
| $0001_B$ | $PCLK/2^1$ |
| $0010_B$ | $PCLK/2^2$ |
| $0011_B$ | $PCLK/2^3$ |
| $0100_B$ | $PCLK/2^4$ |
| $0101_B$ | $PCLK/2^5$ |
| $0110_B$ | $PCLK/2^6$ |
| $0111_B$ | $PCLK/2^7$ |
| $1000_B$ | $PCLK/2^8$ |
| $1001_B$ | $PCLK/2^9$ |
| $1010_B$ | $PCLK/2^{10}$ |
| $1011_B$ | $PCLK/2^{11}$ |
| $1100_B$ | $PCLK/2^{12}$ |
| $1101_B$ | $PCLK/2^{13}$ |
| $1110_B$ | $PCLK/2^{14}$ |
| $1111_B$ | $PCLK/2^{15}$ |

These bits can only be rewritten when all counters using CK0 are stopped (TAUBnTE.TAUBnTEm = 0).

**NOTE**

The TAUBn clock input PCLK is specified in the first part of this section, **Section 24.1.3, Clock Supply**.

## 24.3.3 Details of TAUBn Control Registers

### 24.3.3.1 TAUBnCDRm — TAUBn Channel Data Register

This register functions either as a compare register or as a capture register, depending on the operation mode specified in TAUBnCMORm.TAUBnMD[4:1].

**Access:** This register can be read or written in 16-bit units.
- When this register functions as a capture register, only reading is possible. Write operation is ignored.
- When this register functions as a compare register, reading and writing is possible.

**Address:** <TAUBn_base> + m × $4_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCDR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.12 TAUBnCDRm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnCDR [15:0] | Data register for the capture/compare value. |

### 24.3.3.2 TAUBnCNTm — TAUBn Channel Counter Register

This register is the channel m counter register.

**Access:** This register is a read-only register that can be read in 16-bit units.

**Address:** <TAUBn_base> + $80_H$ + m × $4_H$

**Value after reset:** $FFFF_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCNT[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 24.13 TAUBnCNTm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnCNT [15:0] | 16-bit counter value. |

The read value depends on the counter, the operation mode change, and the values of the TAUBnTS.TAUBnTSm and TAUBnTT.TAUBnTTm bits.

The initial counter read value depends on the operation mode and how the counter was stopped:

- by a reset
- by a counter stop trigger (TAUBnTT.TAUBnTTm = 1)

The following table lists the initial counter read values after the counter has stopped (TAUBnTE.TAUBnTEm = 0) and re-enabled (TAUBnTS.TAUBnTSm = 1).

The table also contains the counter read value one count after the counter is enabled (TAUBnTS.TAUBnTSm = 1) for modes where the counter waits for a start trigger.

**Table 24.14 TAUBnCNTm Read Values after Re-Enabling Counter**

| Mode Name | Count Method (Up/Down) | TAUBnCNTm | | |
|---|---|---|---|---|
| | | Start Value*1 | After Stop Trigger | After One Count |
| Interval timer mode | Count down | $FFFF_H$ | Stop value | — |
| Judge mode | Count down | $FFFF_H$ | Stop value | — |
| Capture mode | Count up | $0000_H$ | Stop value | — |
| Event count mode | Count down | $FFFF_H$ | Stop value | — |
| One-count mode | Count down | $FFFF_H$ | Stop value | Stop value |
| Capture and one-count mode | Count up | $0000_H$ | Stop value | Capture value + 1 (TAUBnCDRm) |
| Judge and one-count mode | Count down | $FFFF_H$ | Stop value | TAUBnCNTm value – 1 |
| Count-up/-down mode | Count up/down | $FFFF_H$ | Stop value | — |
| Pulse one-count mode | Count down | $FFFF_H$ | Stop value | $0000_H$ |
| Count capture mode | Count up | $0000_H$ | Stop value | — |
| Gate count mode | Count down | $FFFF_H$ | Stop value | Stop value |
| Capture and gate count mode | Count up | $0000_H$ | Stop value | Stop value |

Note 1. The value set for TAUBnCNTm when operation mode is changed after reset release

### 24.3.3.3 TAUBnCMORm — TAUBn Channel Mode OS Register

This register controls channel m operation.

**Access:** This register can be read or written in 16-bit units. It can only be written when the counter is stopped (TAUBnTE.TAUBnTEm = 0).

**Address:** <TAUBn_base> + $200_H$ + m × $4_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.15 TAUBnCMORm Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS [1:0] | Selects the operation clock.<br>The operation clock is used for the TAUBTTINm input edge detection circuit. TAUBnCNTm can also be used as the count clock depending on the setting of the TAUBnCMORm.TAUBnCCS0 bit |

| TAUBn CKS1 | TAUBn CKS0 | Selection of Count Clock |
|---|---|---|
| 0 | 0 | CK0 |
| 0 | 1 | CK1 |
| 1 | 0 | CK2 |
| 1 | 1 | CK3 |

| Bit Position | Bit Name | Function |
|---|---|---|
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Selects the count clock for the TAUBnCNTm counter |

| TAUBnCCS0 | Selected Count Clock |
|---|---|
| 0 | Operation clock as specified by TAUBnCMORm.TAUBnCKS[1:0]. |
| 1 | Valid edge of TAUBTTINm input signal |

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 | TAUBnMAS | Specifies the channel as master or slave channel during synchronous channel operation:<br>0: Slave<br>1: Master<br>This bit is only valid for even channels (CHm_even). For odd channels (CHm_odd), it is fixed to 0. |

**Table 24.15 TAUBnCMORm Register Contents (2/3)**

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>10 to 8</td><td>TAUBnSTS [2:0]</td><td>Selects the external start trigger:
<table>
<tr><th>TAUBn STS2</th><th>TAUBn STS1</th><th>TAUBn STS0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Software trigger</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Valid edge of the TAUBTTINm input signal. TAUBnCMURm.TAUBnTIS[1:0] specifies the valid edge.</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Valid edge of the TAUBTTINm input signal is used as the start trigger and the reverse edge is used as the stop trigger.</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Setting prohibited</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>INTTAUBnIm is the start trigger of the master channel</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>INTTAUBnIm of upper channel (m – 1) is the start trigger regardless of master setting</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Dead-time output signal of the TAUBTTOUTm generation unit</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Up/down output trigger signal of the master channel.</td></tr>
</table></td></tr>
<tr><td>7, 6</td><td>TAUBnCOS [1:0]</td><td>Specifies when the capture register TAUBnCDRm and the overflow flag TAUBnCSRm.TAUBnOVF of channel m are updated.<br>These bits are only valid if channel m is in capture function (capture mode and capture & one-count mode).
<table>
<tr><th>TAUBn COS1</th><th>TAUBn COS0</th><th>TAUBnCDRm</th><th>TAUBnCSRm.TAUBnOVF</th></tr>
<tr><td>0</td><td>0</td><td>Updated upon detection of a TAUBTTINm input valid edge.</td><td>Updated (cleared or set) upon detection of a TAUBTTINm input valid edge:<br>• If a counter overflow has occurred since the last valid edge detection, TAUBnCSRm.TAUBnOVF is set.<br>• If no counter overflow has occurred since the last valid edge detection, TAUBnCSRm.TAUBnOVF is cleared.</td></tr>
<tr><td>0</td><td>1</td><td></td><td>Set upon counter overflow and cleared by setting TAUBnCSCm.TAUBnCLOV to 1.</td></tr>
<tr><td>1</td><td>0</td><td rowspan="2">Updated upon detection of a TAUBTTINm input valid edge and upon counter overflow:<br>• TAUBTTINm input valid edge: Counter value is written to TAUBnCDRm<br>• Overflow: $FFFF_H$ is written to TAUBnCDRm. The next TAUBTTINm input valid edge detection is ignored.</td><td>Not set.</td></tr>
<tr><td>1</td><td>1</td><td>Set upon counter overflow and cleared by setting TAUBnCSCm.TAUBnCLOV to 1.</td></tr>
</table></td></tr>
<tr><td>5</td><td>Reserved</td><td>When read, the value after reset is returned. When writing, write the value after reset.</td></tr>
</table>

**Table 24.15 TAUBnCMORm Register Contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 4 to 0 | TAUBnMD [4:0] | Specifies the operation mode. For details, refer to the settings for individual functions. |

| TAUBn MD4 | TAUBn MD3 | TAUBn MD2 | TAUBn MD1 | TAUBn MD0 | Description |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1/0 | Interval timer mode |
| 0 | 0 | 0 | 1 | 1/0 | Judge mode |
| 0 | 0 | 1 | 0 | 1/0 | Capture mode |
| 0 | 0 | 1 | 1 | 0 | Event count mode |
| 0 | 1 | 0 | 0 | 1/0 | One-count mode |
| 0 | 1 | 0 | 1 | 1/0 | Setting prohibited |
| 0 | 1 | 1 | 0 | 0 | Capture and one-count mode |
| 0 | 1 | 1 | 1 | 1/0 | Judge and one-count mode |
| 1 | 0 | 0 | 0 | 0 | Setting prohibited |
| 1 | 0 | 0 | 1 | 0 | Count-up/-down mode |
| 1 | 0 | 1 | 0 | 1/0 | Pulse one-count mode |
| 1 | 0 | 1 | 1 | 1/0 | Count capture mode |
| 1 | 1 | 0 | 0 | 0 | Gate count mode |
| 1 | 1 | 0 | 1 | 0 | Capture and gate count mode |

| Mode | Role of TAUBnMD0 Bit |
|---|---|
| Interval timer mode<br>Capture mode<br>Count capture mode | Specifies whether INTTAUBnIm is output at the beginning of count operation (when a start trigger is entered) or not.<br>0: INTTAUBnIm is not output.<br>1: INTTAUBnIm is output. |
| Event count mode<br>Count-up/-down mode | This bit should be set to 0 (INTTAUBnIm signal is not output at the beginning of count operation). |
| One-count mode<br>Pulse one-count mode | Enables/disables start trigger detection during counting.<br>0: Disables detection.<br>1: Enables detection.<br>**CAUTION**<br>• In one-count mode, INTTAUBnIm signal is not output at the beginning of count operation.<br>• In pulse one-count mode, INTTAUBnIm signal is output at the beginning of count operation. |
| Gate count mode | This bit should be set to 0 (start trigger detection during counting is disabled). |
| Capture and one-count mode<br>Capture and gate count mode | This bit should be set to 0.<br>**CAUTION**<br>INTTAUBnIm signal is not output at the beginning of count operation. In addition, start trigger detected during counting is disabled. |
| Judge mode<br>Judge and one-count mode | Specifies INTTAUBnIm output timing.<br>0: When TAUBnCNTm ≤ TAUBnCDRm<br>1: When TAUBnCNTm > TAUBnCDRm |

### 24.3.3.4 TAUBnCMURm — TAUBn Channel Mode User Register

This register specifies the type of valid edge detection used for the TAUBTTINm input.

**Access:** This register can be read or written in 8-bit units.

**Address:** <TAUBn_base> + $C0_H$ + m × $4_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.16 TAUBnCMURm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS [1:0] | Specifies the valid edge of the TAUBTTINm input signal: |

| TAUBn TIS1 | TAUBn TIS0 | Description |
|---|---|---|
| 0 | 0 | Falling edge |
| 0 | 1 | Rising edge |
| 1 | 0 | Rising and falling edges (low-width measurement selection).<br>Start trigger: falling edge<br>Stop trigger (capture): rising edge |
| 1 | 1 | Rising and falling edges (high-width measurement selection).<br>Start trigger: rising edge<br>Stop trigger (capture): falling edge |

- Edge detection for TAUBTTINm input signals is performed based on the operation clock selected by TAUBnCMORm.TAUBnCKS[1:0].

### 24.3.3.5 TAUBnCSRm — TAUBn Channel Status Register

This register indicates the count direction and the overflow status of the counter for channel m.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <TAUBn_base> + $140_H$ + m × $4_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnCSF | TAUBnOVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 24.17 TAUBnCSRm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. |
| 1 | TAUBnCSF | Indicates the count direction:<br>0: Counts up<br>1: Counts down<br>The read value of this bit is only valid in the following mode:<br>• Up/Down Count mode |
| 0 | TAUBnOVF | Indicates the counter overflow status:<br>0: No overflow occurred<br>1: Overflow occurred<br>This bit is used only in the following modes:<br>• Capture mode<br>• Capture and one-count mode<br>The function of this bit depends on the setting of control bits TAUBnCMORm.TAUBnCOS[1:0]. |

### 24.3.3.6 TAUBnCSCm — TAUBn Channel Status Clear Register

This register is a trigger register for clearing the overflow flag TAUBnCSRm.TAUBnOVF of channel m.

**Access:** This register is a write-only register that can be written in 8-bit units. It is always read as $00_H$

**Address:** <TAUBn_base> + $180_H$ + m × $4_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | TAUBnCLOV |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 24.18 TAUBnCSCm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | TAUBnCLOV | 0: No function<br>1: Clears the overflow flag TAUBnCSRm.TAUBnOVF |

### 24.3.3.7 TAUBnTS — TAUBn Channel Start Trigger Register

This register enables the counter for each channel.

**Access:** This register is a write-only register that can be written in 16-bit units. It is always read as $0000_H$

**Address:** <TAUBn_base> + $1C4_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnTS15 | TAUBnTS14 | TAUBnTS13 | TAUBnTS12 | TAUBnTS11 | TAUBnTS10 | TAUBnTS09 | TAUBnTS08 | TAUBnTS07 | TAUBnTS06 | TAUBnTS05 | TAUBnTS04 | TAUBnTS03 | TAUBnTS02 | TAUBnTS01 | TAUBnTS00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 24.19 TAUBnTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnTSm | Enables the counter for channel m:<br>0: No function<br>1: Enables the counter and sets TAUBnTE.TAUBnTEm = 1.<br>TAUBnTE.TAUBnTEm = 1 only enables the counter. Whether the counter starts depends on the selected operation mode. |

### 24.3.3.8 TAUBnTE — TAUBn Channel Enable Status Register

This register indicates whether counter is enabled or disabled.

**Access:** This register is a read-only register that can be read in 16-bit units.

**Address:** <TAUBn_base> + $1C0_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnTE15 | TAUBnTE14 | TAUBnTE13 | TAUBnTE12 | TAUBnTE11 | TAUBnTE10 | TAUBnTE09 | TAUBnTE08 | TAUBnTE07 | TAUBnTE06 | TAUBnTE05 | TAUBnTE04 | TAUBnTE03 | TAUBnTE02 | TAUBnTE01 | TAUBnTE00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 24.20 TAUBnTE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnTEm | Indicates whether counter for channel m is enabled or disabled:<br>0: Counter disabled<br>1: Counter enabled<br>Setting TAUBnTS.TAUBnTSm to 1 sets this bit to 1.<br>Setting TAUBnTT.TAUBnTTm to 1 resets this bit to 0. |

### 24.3.3.9 TAUBnTT — TAUBn Channel Stop Trigger Register

This register stops the counter for each channel.

**Access:** This register is a write-only register that can be written in 16-bit units. It is always read as $0000_H$.

**Address:** <TAUBn_base> + $1C8_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBn TT15 | TAUBn TT14 | TAUBn TT13 | TAUBn TT12 | TAUBn TT11 | TAUBn TT10 | TAUBn TT09 | TAUBn TT08 | TAUBn TT07 | TAUBn TT06 | TAUBn TT05 | TAUBn TT04 | TAUBn TT03 | TAUBn TT02 | TAUBn TT01 | TAUBn TT00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 24.21 TAUBnTT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnTTm | Stops the counter of channel m:<br>0: No function<br>1: Stops the counter and resets TAUBnTE.TAUBnTEm.<br>TAUBnCNTm, TAUBnTO.TAUBnTOm, and TAUBTTOUTm all retain the values they had before the counter was stopped. |

## 24.3.4 Details of TAUBn Simultaneous Rewrite Registers

### 24.3.4.1 TAUBnRDE — TAUBn Channel Reload Data Enable Register

This register enables and disables simultaneous rewrite of the data register TAUBnCDRm/ TAUBnTOLm.

**Access:** This register can be read or written in 16-bit units. It can only be written when TAUBnTE.TAUBnTEm = 0.

**Address:** <TAUBn_base> + $260_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBn RDE15 | TAUBn RDE14 | TAUBn RDE13 | TAUBn RDE12 | TAUBn RDE11 | TAUBn RDE10 | TAUBn RDE09 | TAUBn RDE08 | TAUBn RDE07 | TAUBn RDE06 | TAUBn RDE05 | TAUBn RDE04 | TAUBn RDE03 | TAUBn RDE02 | TAUBn RDE01 | TAUBn RDE00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.22 TAUBnRDE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnRDEm | Enables/disables simultaneous rewrite of the data register of channel m:<br>0: Disables simultaneous rewrite<br>1: Enabled simultaneous rewrite |

### 24.3.4.2 TAUBnRDS — TAUBn Channel Reload Data Control Channel Select Register

This register selects the control channel for simultaneous rewrite.

**Access:** This register can be read or written in 16-bit units. It can only be written when TAUBnTE.TAUBnTEm = 0.

**Address:** <TAUBn_base> + $268_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBn RDS15 | TAUBn RDS14 | TAUBn RDS13 | TAUBn RDS12 | TAUBn RDS11 | TAUBn RDS10 | TAUBn RDS09 | TAUBn RDS08 | TAUBn RDS07 | TAUBn RDS06 | TAUBn RDS05 | TAUBn RDS04 | TAUBn RDS03 | TAUBn RDS02 | TAUBn RDS01 | TAUBn RDS00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.23 TAUBnRDS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnRDSm | Specifies which channel is controlled for the simultaneous rewrite trigger:<br>0: Master channel<br>1: Another upper channel |

### 24.3.4.3 TAUBnRDM — TAUBn Channel Reload Data Mode Register

This register selects when the signal that controls simultaneous rewrite is generated.

**Access:** This register can be read or written in 16-bit units. It can only be written when TAUBnTE.TAUBnTEm = 0.

**Address:** <TAUBn_base> + $264_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBn RDM15 | TAUBn RDM14 | TAUBn RDM13 | TAUBn RDM12 | TAUBn RDM11 | TAUBn RDM10 | TAUBn RDM09 | TAUBn RDM08 | TAUBn RDM07 | TAUBn RDM06 | TAUBn RDM05 | TAUBn RDM04 | TAUBn RDM03 | TAUBn RDM02 | TAUBn RDM01 | TAUBn RDM00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.24 TAUBnRDM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnRDMm | Selects when the signal that triggers simultaneous rewrite is generated:<br>0: When the master channel counter starts counting<br>1: At the top of a triangle wave cycle<br>These bits only apply when TAUBnRDE.TAUBnRDEm = 1 and TAUBnRDS.TAUBnRDSm = 0. |

### 24.3.4.4 TAUBnRDC — TAUBn Channel Reload Data Control Register

This register specifies the channel that generates the INTTAUBnIm signal that triggers simultaneous rewrite.

**Access:** This register can be read or written in 16-bit units. It can only be written when TAUBnTE.TAUBnTEm = 0

**Address:** <TAUBn_base> + $26C_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBn RDC15 | TAUBn RDC14 | TAUBn RDC13 | TAUBn RDC12 | TAUBn RDC11 | TAUBn RDC10 | TAUBn RDC09 | TAUBn RDC08 | TAUBn RDC07 | TAUBn RDC06 | TAUBn RDC05 | TAUBn RDC04 | TAUBn RDC03 | TAUBn RDC02 | TAUBn RDC01 | TAUBn RDC00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.25 TAUBnRDC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnRDCm | Specifies whether the channel generates a simultaneous rewrite trigger signal or not.<br>0: Does not operate as a simultaneous rewrite trigger channel.<br>1: Operates as a simultaneous rewrite trigger channel.<br>These bits only apply when TAUBnRDE.TAUBnRDEm = 1 and TAUBnRDS.TAUBnRDSm = 1. |

### 24.3.4.5 TAUBnRDT — TAUBn Channel Reload Data Trigger Register

This register triggers the simultaneous rewrite enabling state.

**Access:** This register is a write-only register that can be written in 16-bit units. It is always read as $0000_H$.

**Address:** <TAUBn_base> + $044_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBn RDT15 | TAUBn RDT14 | TAUBn RDT13 | TAUBn RDT12 | TAUBn RDT11 | TAUBn RDT10 | TAUBn RDT09 | TAUBn RDT08 | TAUBn RDT07 | TAUBn RDT06 | TAUBn RDT05 | TAUBn RDT04 | TAUBn RDT03 | TAUBn RDT02 | TAUBn RDT01 | TAUBn RDT00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 24.26 TAUBnRDT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnRDTm | Triggers the simultaneous rewrite enabling state:<br>0: No function. Writing 0 is ignored (the operation is not affected).<br>1: The simultaneous rewrite enabling flag (TAUBnRSFm) is set to 1. The system waits for the simultaneous rewrite trigger.<br>These bits only apply when:<br>• TAUBnRDE.TAUBnRDEm = 1 |

### 24.3.4.6 TAUBnRSF — TAUBn Channel Reload Status Register

This flag register indicates the simultaneous rewrite status.

**Access:** This register is a read-only register that can be read in 16-bit units.

**Address:** <TAUBn_base> + $048_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBn RSF15 | TAUBn RSF14 | TAUBn RSF13 | TAUBn RSF12 | TAUBn RSF11 | TAUBn RSF10 | TAUBn RSF09 | TAUBn RSF08 | TAUBn RSF07 | TAUBn RSF06 | TAUBn RSF05 | TAUBn RSF04 | TAUBn RSF03 | TAUBn RSF02 | TAUBn RSF01 | TAUBn RSF00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 24.27 TAUBnRSF Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnRSFm | Indicates the simultaneous rewrite status:<br>0: Indicates simultaneous rewrite is completed due to the generation of the simultaneous rewrite trigger.<br>1: Indicates the simultaneous rewrite trigger waiting state when simultaneous rewrite is enabled (TAUBnRDTm = 1). |

## 24.3.5 Details of TAUBn Output Registers

### 24.3.5.1 TAUBnTOE — TAUBn Channel Output Enable Register

This register enables and disables independent channel output mode controlled by software.

**Access:** This register can be read or written in 16-bit units.

**Address:** <TAUBn_base> + $5C_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBn TOE15 | TAUBn TOE14 | TAUBn TOE13 | TAUBn TOE12 | TAUBn TOE11 | TAUBn TOE10 | TAUBn TOE09 | TAUBn TOE08 | TAUBn TOE07 | TAUBn TOE06 | TAUBn TOE05 | TAUBn TOE04 | TAUBn TOE03 | TAUBn TOE02 | TAUBn TOE01 | TAUBn TOE00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.28 TAUBnTOE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnTOEm | Enables/disables independent channel output mode:<br>0: Disables independent channel output mode (controlled by software)<br>1: Enables independent channel output mode |

### 24.3.5.2 TAUBnTO — TAUBn Channel Output Register

This register specifies and reads the level of TAUBTTOUTm.

**Access:** This register can be read or written in 16-bit units.

**Address:** <TAUBn_base> + $58_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBn TO15 | TAUBn TO14 | TAUBn TO13 | TAUBn TO12 | TAUBn TO11 | TAUBn TO10 | TAUBn TO09 | TAUBn TO08 | TAUBn TO07 | TAUBn TO06 | TAUBn TO05 | TAUBn TO04 | TAUBn TO03 | TAUBn TO02 | TAUBn TO01 | TAUBn TO00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.29 TAUBnTO Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnTOm | Specifies/reads the level of TAUBTTOUTm:<br>0: Low<br>1: High<br>Only TAUBnTOm bits for which Independent Channel Output function is disabled (TAUBnTOEm = 0) can be written. |

### 24.3.5.3 TAUBnTOM — TAUBn Channel Output Mode Register

This register specifies the output mode of each channel.

**Access:** This register can be read or written in 16-bit units. It can only be written when the counter is stopped (TAUBnTE.TAUBnTEm = 0).

**Address:** <TAUBn_base> + $248_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBn TOM15 | TAUBn TOM14 | TAUBn TOM13 | TAUBn TOM12 | TAUBn TOM11 | TAUBn TOM10 | TAUBn TOM09 | TAUBn TOM08 | TAUBn TOM07 | TAUBn TOM06 | TAUBn TOM05 | TAUBn TOM04 | TAUBn TOM03 | TAUBn TOM02 | TAUBn TOM01 | TAUBn TOM00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.30 TAUBnTOM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnTOMm | Specifies the channel output mode:<br>0: Independent channel output mode<br>1: Synchronous channel output mode |

### 24.3.5.4 TAUBnTOC — TAUBn Channel Output Configuration Register

This register specifies the output mode of each channel in combination with TAUBnTOMm.

**Access:** This register can be read or written in 16-bit units. It can only be written when the counter is stopped (TAUBnTE.TAUBnTEm = 0).

**Address:** <TAUBn_base> + $24C_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBn TOC15 | TAUBn TOC14 | TAUBn TOC13 | TAUBn TOC12 | TAUBn TOC11 | TAUBn TOC10 | TAUBn TOC09 | TAUBn TOC08 | TAUBn TOC07 | TAUBn TOC06 | TAUBn TOC05 | TAUBn TOC04 | TAUBn TOC03 | TAUBn TOC02 | TAUBn TOC01 | TAUBn TOC00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.31 TAUBnTOC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnTOCm | Specifies the output mode:<br>0: Operation mode 1<br>1: Operation mode 2<br>The output mode also depends on TAUBnTOM.TAUBnTOMm, as can be seen in the following table. |

| TOMm | TOCm | Description |
|---|---|---|
| 0 | 0 | Toggle mode: Toggles when INTTAUBnIm occurs. |
| | 1 | Set/reset mode: Set when INTTAUBnIm occurs upon count start and reset when INTTAUBnIm occurs due to detection of a match between TAUBnCNTm and TAUBnCDRm. |
| 1 | 0 | Synchronous channel operation mode 1:<br>Set when INT occurs on the master channel and reset when INT occurs on the slave channel. |
| | 1 | Synchronous channel operation mode 2:<br>Set when INTTAUBnIm occurs while the slave channel is counting down and reset when INTTAUBnIm occurs while the slave channel is counting up. |

### 24.3.5.5 TAUBnTOL — TAUBn Channel Output Active Level Register

This register specifies the output logic of the channel output bit (TAUBnTO.TAUBnTOm).

**Access:** This register can be read or written in 16-bit units.

**Address:** <TAUBn_base> + $040_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBn TOL15 | TAUBn TOL14 | TAUBn TOL13 | TAUBn TOL12 | TAUBn TOL11 | TAUBn TOL10 | TAUBn TOL09 | TAUBn TOL08 | TAUBn TOL07 | TAUBn TOL06 | TAUBn TOL05 | TAUBn TOL04 | TAUBn TOL03 | TAUBn TOL02 | TAUBn TOL01 | TAUBn TOL00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.32 TAUBnTOL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnTOLm | Specifies the output logic of the channel m output bit (TAUBnTO.TAUBnTOm):<br>0: Positive logic (active high)<br>1: Negative logic (active low)<br>The setting of these bits applies to all channel output modes other than independent channel output mode controlled by software. |

## 24.3.6 Details of TAUBn Dead Time Output Registers

### 24.3.6.1 TAUBnTDE — TAUBn Channel Dead Time Output Enable Register

This register enables/disables dead time operation for each channel.

**Access:** This register can be read or written in 16-bit units. It can only be written when the counter is stopped (TAUBnTE.TAUBnTEm = 0).

**Address:** <TAUBn_base> + $250_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBn TDE15 | TAUBn TDE14 | TAUBn TDE13 | TAUBn TDE12 | TAUBn TDE11 | TAUBn TDE10 | TAUBn TDE09 | TAUBn TDE08 | TAUBn TDE07 | TAUBn TDE06 | TAUBn TDE05 | TAUBn TDE04 | TAUBn TDE03 | TAUBn TDE02 | TAUBn TDE01 | TAUBn TDE00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.33 TAUBnTDE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnTDEm | Enables/disables dead time control operation of channel m:<br>0: Disables dead time operation<br>1: Enables dead time operation<br>The same settings must be set for the even and the odd slave channel that comprise a set.<br>These bits only apply when:<br>• TAUBnTOE.TAUBnTOEm, TAUBnTOM.TAUBnTOMm, TAUBnTOC.TAUBnTOCm = 1 |

### 24.3.6.2 TAUBnTDL — TAUBn Channel Dead Time Output Level Register

This register selects the phase period to which dead time is added.

**Access:** This register can be read or written in 16-bit units.

**Address:** <TAUBn_base> + $54_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBn TDL15 | TAUBn TDL14 | TAUBn TDL13 | TAUBn TDL12 | TAUBn TDL11 | TAUBn TDL10 | TAUBn TDL09 | TAUBn TDL08 | TAUBn TDL07 | TAUBn TDL06 | TAUBn TDL05 | TAUBn TDL04 | TAUBn TDL03 | TAUBn TDL02 | TAUBn TDL01 | TAUBn TDL00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.34 TAUBnTDL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUBnTDLm | Selects the phase period to which dead time is added:<br>0: Positive phase period<br>1: Negative phase period<br>These bits only apply when:<br>• TAUBnTOE.TAUBnTOEm, TAUBnTOM.TAUBnTOMm, TAUBnTOC.TAUBnTOCm, TAUBnTDE.TAUBnTDEm = 1 |

## 24.3.7 TAUBn Emulation Register

### 24.3.7.1 TAUBnEMU — TAUBn Emulation Register

This register controls SVSTOP operations.

**Access:** This register can be read or written in 8-bit units.
Write to this register only when the counter is stopped (TAUBnTE.TAUBnTEm = 0) and when EPC.SVSTOP = 0.

**Address:** <TAUBn_base> + $290_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TAUBnSVSDIS | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R |

**Table 24.35 TAUBnEMU Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | TAUBnSVSDIS | (When EPC.SVSTOP = 0)<br>Regardless of the value of this bit (1/0), the count clock is continuously supplied when the debugger obtains the control of the microcontroller (e.g., at a breakpoint).<br><br>(When EPC.SVSTOP = 1)<br>0: The count clock stops when the debugger obtains the control of the microcontroller (e.g., at a breakpoint).<br>1: The count clock is continuously supplied when the debugger obtains the control of the microcontroller (e.g., at a breakpoint). |
| 6 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

## 24.4 General Operating Procedure

The following describes the general operation procedure for the TAUBn:

After reset release, the operation of each channel is stopped. Clock supply is started and writing to each register is enabled. All circuits and registers of all channels are initialized. The control register of TAUBTTOUTm is also initialized and outputs a low level.

1. Set the TAUBnTPS register to specify the clock frequency of CK0 to CK3.
2. Configure the desired TAUBn function:
   - Set the operation mode
   - Set the channel output mode
   - Set any other control bits
3. Enable the counter by setting the TAUBnTS.TAUBnTSm bit to 1.
   The counter starts to count immediately, or when an appropriate trigger is detected, depending on the bit settings.
4. If desired, and if possible for the configured function, stop the counter or perform a forced restart operation during count operation. The counter can be stopped by setting the TAUBnTT.TAUBnTTm bit to 1. The counter can be forcibly restarted by setting the TAUBnTS.TAUBnTSm bit to 1.
5. Stop the function by setting the TAUBnTT.TAUBnTTm bit to 1.

**NOTE**

- A detailed description of the required control bits and the operation of the individual functions is given below.
  - **Section 24.12, Independent Channel Operation Functions**
  - **Section 24.14, Synchronous Channel Operation Functions**
- The function can be changed while the counter is stopped (TAUBnTE.TAUBnTEm=0).

## 24.5 Concepts of Synchronous Channel Operation

The synchronous channel operation function is implemented using a combination of channel groups (consisted of master and slave channels).

Several rules apply to the settings of channels.

These rules are detailed in **Section 24.5.1, Rules of Synchronous Channel Operation Function**.

Two special features for synchronous channel operation are detailed in the following:

- **Section 24.5.2, Simultaneous Start and Stop of Synchronous Channel Counters**
- **Section 24.6, Simultaneous Rewrite**

### 24.5.1 Rules of Synchronous Channel Operation Function

#### Number of master and slave channels

- Only even channels (CH0, CH2, CH4, …) can be set as master channels.
  Any channel apart from CH0 can be set as a slave channel.
- Only channels lower than the master channel can be set as slave channels, and multiple slave channels can be set for one master channel.
  Example: If CH2 is a master channel, CH3 and the lower channels (CH3, CH4, CH5, …) can be set as slave channels.
- If multiple master channels are used, a range of slave channels that includes another master channel cannot be set for a given master channel.
  Example: If CH0 and CH4 are master channels, CH1 to CH3 can be set as slave channels for CH0, but CH5 to CH15 cannot.

#### Operation clock

- The same operation clock must be set for the master channel and the synchronized slave channel. This is achieved by setting the same value to the TAUBnCMORm.TAUBnCKS[1:0] bits of the master and slave channels.

The basic concepts of master/slave channel usage and operation clocks are illustrated in **Figure 24.3**.

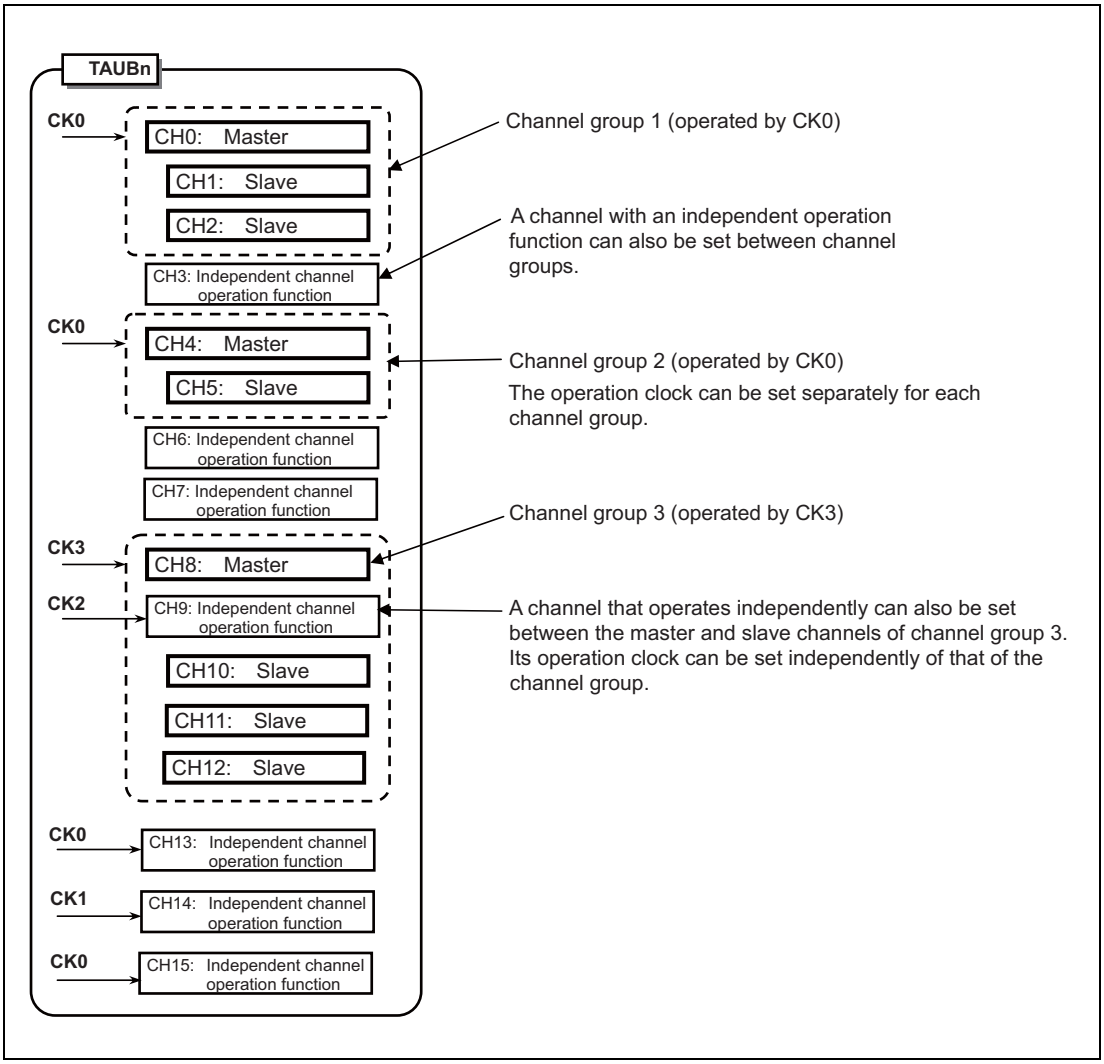



**Figure 24.3 Grouping of the Channels and Assignment of Operation Clocks**

### 24.5.2 Simultaneous Start and Stop of Synchronous Channel Counters

Channels that are operated synchronously can be started and stopped simultaneously within the same unit.

#### 24.5.2.1 Simultaneous Start and Stop within the Same Unit

- To simultaneously start synchronized channels, the TAUBnTS.TAUBnTSm bits of the channels must be set at the same time.
- To simultaneously stop synchronized channels, the TAUBnTT.TAUBnTTm bits of the channels must be set at the same time.

Setting the TAUBnTS.TAUBnTSm bits to 1 sets the corresponding TAUBnTE.TAUBnTEm bits to 1, enabling counting. The exact time that it starts depends on the operation mode.

## 24.6 Simultaneous Rewrite

### 24.6.1 Introduction

Simultaneous rewrite describes the ability to change the compare/start value and the output logic of multiple channels at the same time.

The corresponding data and control registers (TAUBnCDRm and TAUBnTOLm) can nevertheless be written at any time. The new value does not affect the counter operation or the output signal until simultaneous rewrite is triggered.

Simultaneous rewrite can be triggered by:

- The counter on the master channel or upper channel (depending on the selected operation mode) reaching a certain value
- INTTAUBnIm being issued on the upper channel specified by TAUBnRDC.TAUBnRDCm

There are three methods for simultaneous rewrite. These are listed in the following table, along with how to specify them and when they cause simultaneous rewrite to be triggered.

**Table 24.36 Simultaneous Rewrite Methods and when They are Triggered**

| Method | Trigger | TAUBnRDE.TAUBnRDEm | TAUBnRDS.TAUBnRDSm | TAUBnRDM.TAUBnRDMm |
|---|---|---|---|---|
| — | No simultaneous rewrite | 0 | 0 | 0 |
| A | The master channel (re)starts counting | 1 | 0 | 0 |
| B | Counting is started in the master channel. The master channel starts counting down at the peak of triangular wave of the corresponding slave channel. | 1 | 0 | 1 |
| C1 | INTTAUBnIm is generated on an upper channel specified by TAUBnRDC.TAUBnRDCm | 1 | 1 | 0/1 |

The following table lists which of these three methods is available for each channel operation function. For more information about the individual channel operation functions, see the corresponding sections in **Section 24.13, Independent Channel Simultaneous Rewrite Functions** and **Section 24.14, Synchronous Channel Operation Functions**.

**Table 24.37 Channel Functions and the Methods They Use for Simultaneous Rewrite**

| Descriptions | A | B | C1 | TAUBnTOL.TAUBnTOLm |
|---|---|---|---|---|
| Simultaneous Rewrite Trigger Output Function Type 1 | | | √ | |
| PWM Output Function | √ | | √ | √ |
| One-Shot Pulse Output Function | √ | | | |
| Delay Pulse Output Function | √ | | | |
| Triangle PWM Output Function | | √ | √ | √ |
| Triangle PWM Output Function with Dead Time | | √ | √ | |
| A/D Conversion Trigger Output Function Type 1 | √ | | √ | |
| A/D Conversion Trigger Output Function Type 2 | | √ | √ | |

**Note:** √: Available, (Blank): Unavailable

### 24.6.2 How to Control Simultaneous Rewrite

The following figure shows the general procedure for simultaneous rewrite.
The three main blocks (Initial settings, Start counter & count operation, and Simultaneous rewrite) are explained afterwards.

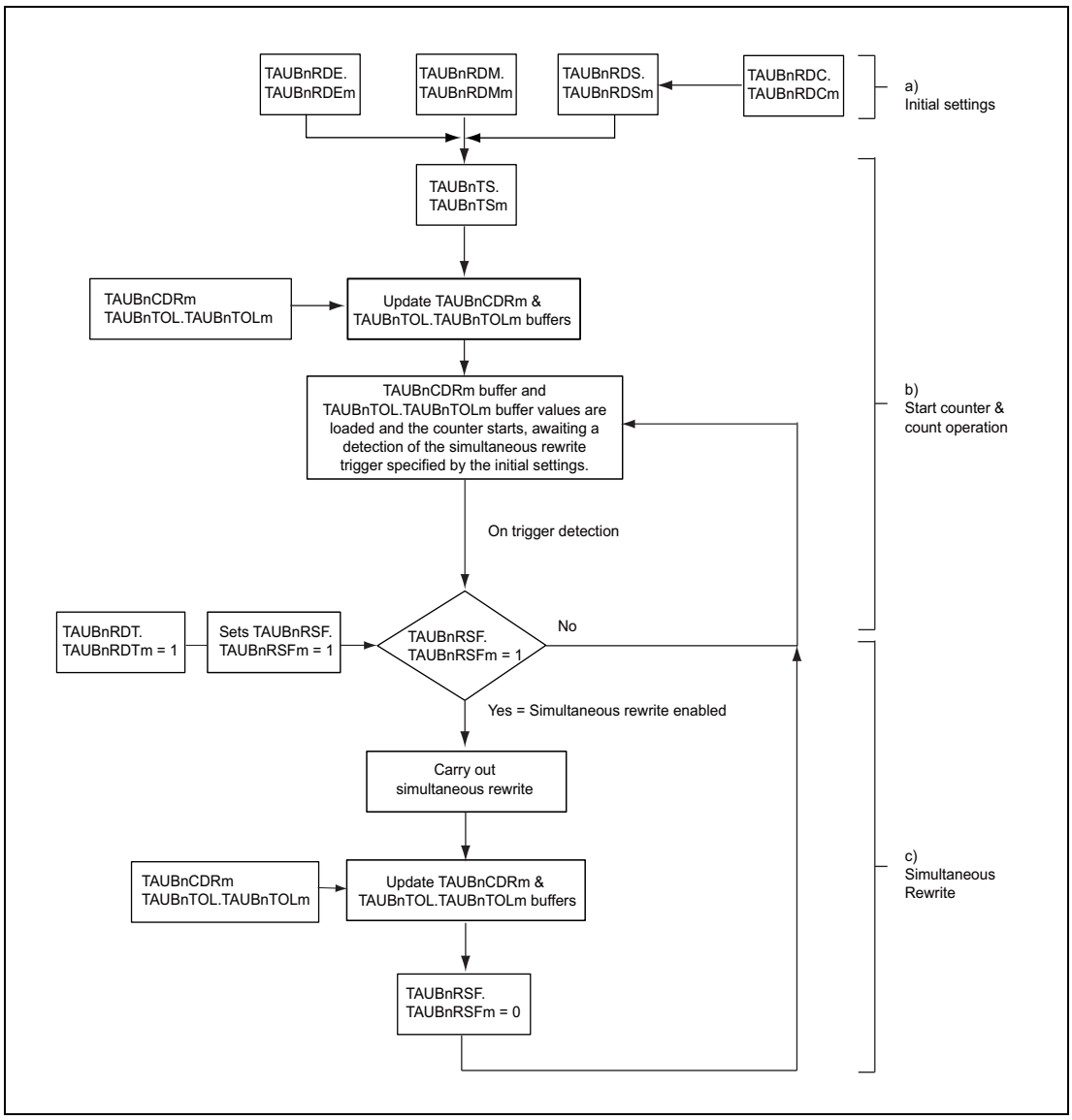



**Figure 24.4 General Procedure for Simultaneous Rewrite**

#### 24.6.2.1 Initial Settings

- To enable simultaneous rewrite in channel m, set TAUBnRDE.TAUBnRDEm = 1.
- To select the type of simultaneous rewrite, set TAUBnRDM.TAUBnRDMm and TAUBnRDS.TAUBnRDSm according to the values in **Table 24.36, Simultaneous Rewrite Methods and when They are Triggered**.
- Specify a simultaneous rewrite trigger channel by using TAUBnRDC.TAUBnRDCm. (Prerequisite: TAUBnRDS.TAUBnRDSm has been set to the upper channel.)

#### 24.6.2.2 Start Counter and Count Operation

- To start all the TAUBnCNTm counters in the channel group, set the corresponding TAUBnTS.TAUBnTSm bits to 1. TAUBnTOL.TAUBnTOLm and the values in the data registers (TAUBnCDRm) are written to the corresponding TAUBnTOL.TAUBnTOLm buffer (TAUBnTOL.TAUBnTOLm buf) and data buffer registers (TAUBnCDRm buf) and the counters start.
- Setting the reload data trigger bit (TAUBnRDT.TAUBnRDTm) to 1 sets the reload flag (TAUBnRSF.TAUBnRSFm) to 1, enabling simultaneous rewrite. TAUBnRSF.TAUBnRSFm remains at 1 until simultaneous rewrite has taken place.
- When the specified trigger for simultaneous rewrite is detected, the TAUBnRSF.TAUBnRSFm bit is checked to see if simultaneous rewrite is enabled (TAUBnRSF.TAUBnRSFm = 1). If it is, simultaneous rewrite is carried out.
  Otherwise, simultaneous rewrite is not carried out, and the system awaits the next simultaneous rewrite trigger detection.

#### 24.6.2.3 Simultaneous Rewrite

- When simultaneous rewrite is enabled (TAUBnRSF.TAUBnRSFm = 1) and the simultaneous rewrite trigger is detected, the current values of the data registers are copied to their buffers. These values are then written to the corresponding counters and the values are applied the next time the counter starts or restarts.
- When simultaneous rewrite is finished, the TAUBnRSF.TAUBnRSFm bit is set to 0, and the system awaits the next simultaneous rewrite trigger.

### 24.6.3 Other General Rules of Simultaneous Rewrite

The following rules also apply:

- TAUBnRDE.TAUBnRDEm, TAUBnRDS.TAUBnRDSm, TAUBnRDM.TAUBnRDMm, and TAUBnRDC.TAUBnRDCm cannot be changed while the counter is in operation (TAUBnTE.TAUBnTEm = 1).
- TAUBnTOL.TAUBnTOLm can only be rewritten during operation in PWM output function or triangle PWM output function. For all other output functions, TAUBnTOL.TAUBnTOLm must be written before the counter starts. If it is rewritten in another function mode, TAUBTTOUTm outputs an invalid wave.
- When a simultaneous rewrite trigger is issued on an upper channel (TAUBnRDS.TAUBnRDSm = 1), the TAUBnRDC.TAUBnRDCm bit controls all the lower channels. This means that if the TAUBnRDC.TAUBnRDCm bits of CH2 and CH7 are set to 1 and the TAUBnRDC.TAUBnRDCm bits of other channels are set to 0, CH2 and CH7 serve as simultaneous rewrite trigger generation channels. CH2 controls the lower channels CH3 to CH6, and CH7 controls the lower channels CH8 to CH15.
- If simultaneous rewrite is enabled and an upper channel is selected for the simultaneous rewrite trigger generation channel (TAUBnRDE.TAUBnRDEm and TAUBnRDS.TAUBnRDSm = 1) but no upper channel is set (TAUBnRDC.TAUBnRDC[15:0] = 0), simultaneous rewrite cannot take place.

### 24.6.4 Types of Simultaneous Rewrite

In the following section, the three simultaneous rewrite methods are explained using timing diagrams.

#### 24.6.4.1 Simultaneous Rewrite when the Master Channel (Re)Starts Counting (Method A)

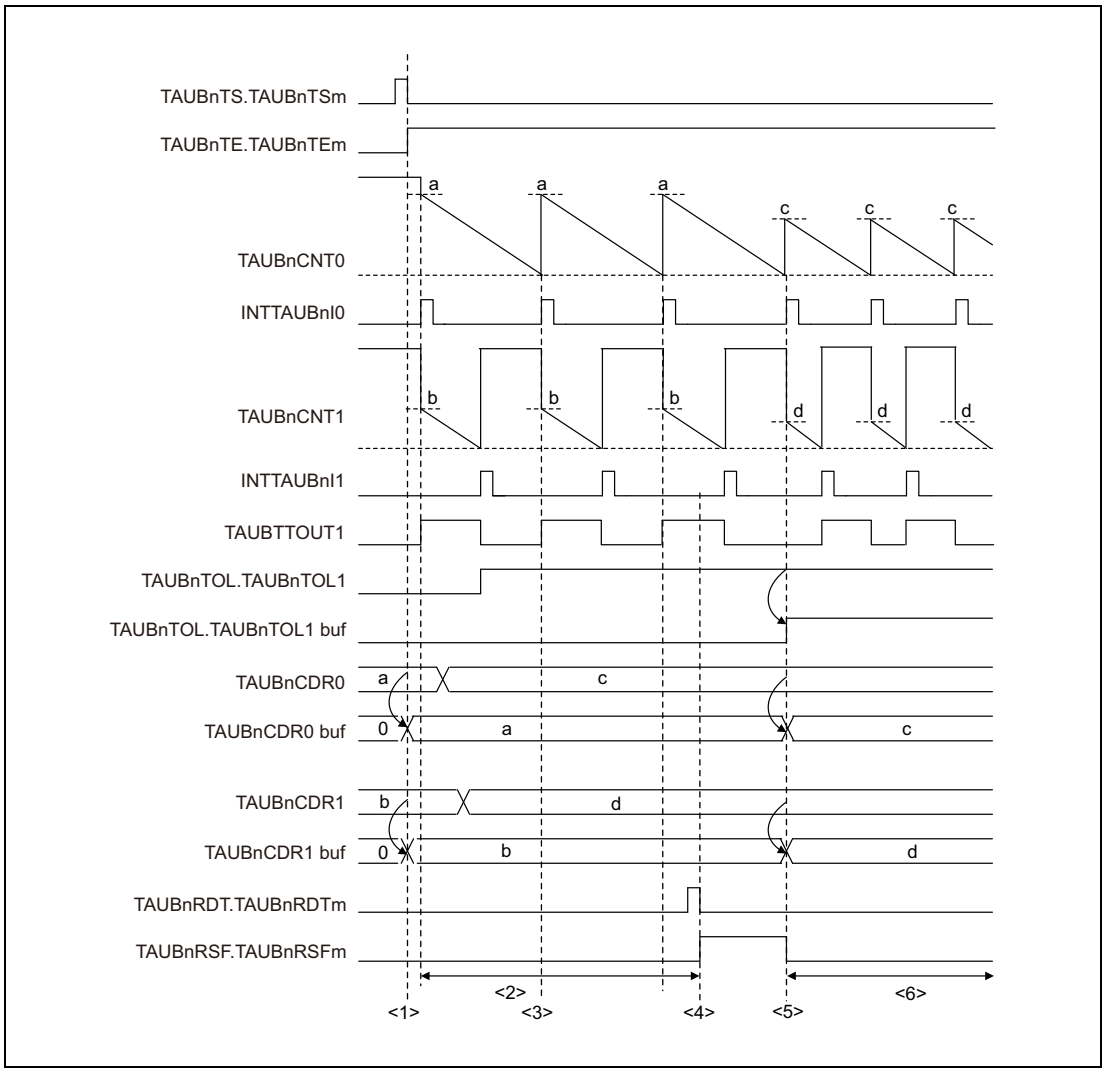



Figure 24.5 Simultaneous Rewrite when the Master Channel (Re)Starts Counting

**Setting:**

CH0 is the master channel, which starts counting down, and CH1 represents an arbitrary slave channel. The simultaneous rewrite method A is applied.

**Description:**

(1) When TAUBnTS.TAUBnTSm = 1 is set, the value of TAUBnCDRm is copied to the TAUBnCDRm buffer and the value of TAUBnTOL.TAUBnTOLm is copied to the TAUBnTOL.TAUBnTOLm buffer.

(2) The TAUBnCDRm and TAUBnTOL.TAUBnTOLm registers can be written at any time.

(3) CH0 restarts counting, but simultaneous rewrite does not occur because it is disabled (TAUBnRSF.TAUBnRSFm = 0)

(4) The reload data trigger bit (TAUBnRDT.TAUBnRDTm) is set to 1 which sets the status flag (TAUBnRSF.TAUBnRSFm = 1), enabling simultaneous rewrite.

(5) Because simultaneous rewrite is enabled, it is triggered when CH0 restarts counting. The TAUBnCDRm value is loaded into the TAUBnCDRm buffer, and the TAUBnTOL.TAUBnTOLm value is loaded into the TAUBnTOL.TAUBnTOLm buffer.

(6) The counters count down and await the next simultaneous rewrite trigger. The values of TAUBnCDRm and TAUBnTOL.TAUBnTOLm can be changed again.

### 24.6.4.2 Simultaneous Rewrite at the Peak of a Triangular Wave of the Slave Channel (Method B)

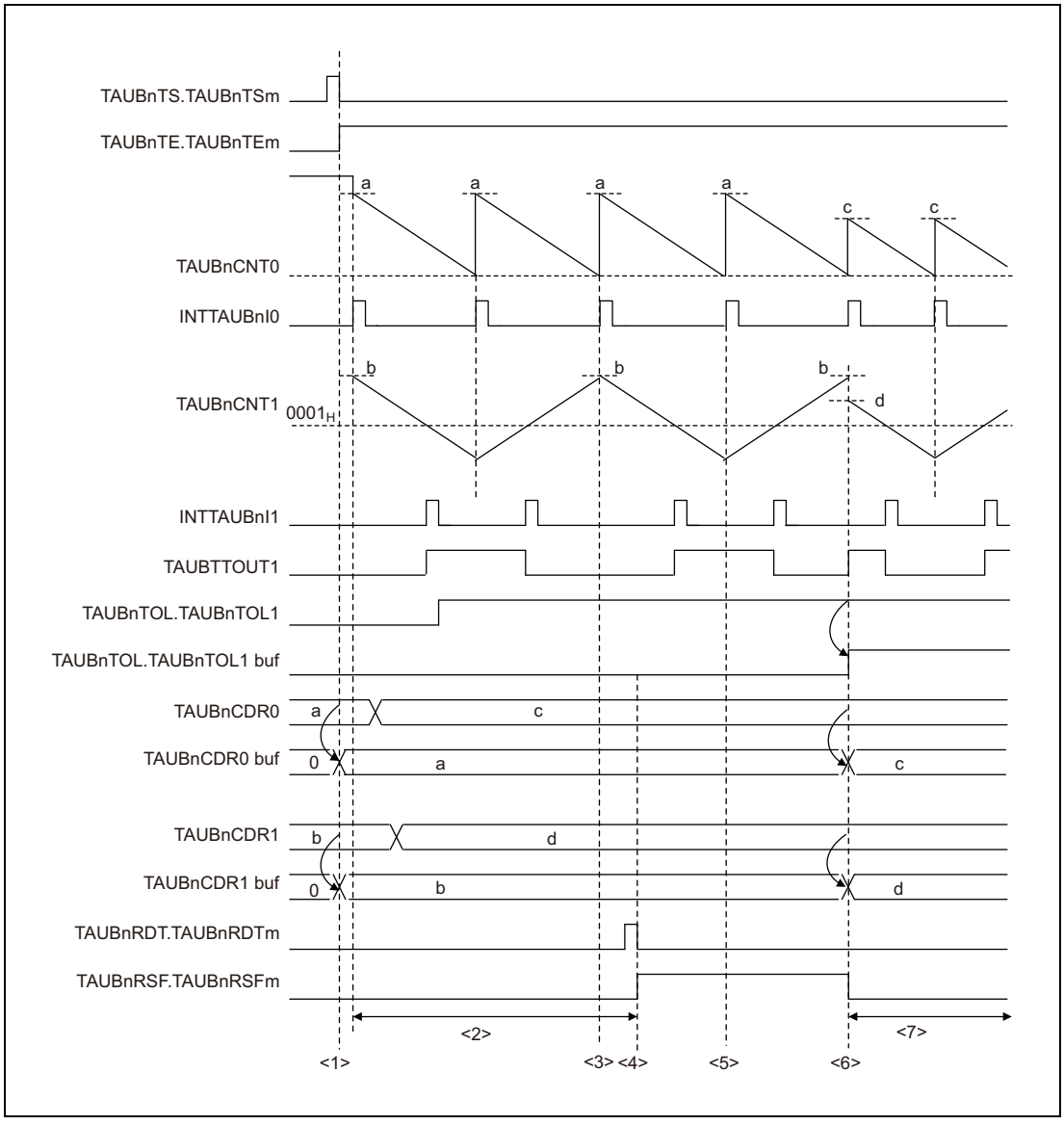



**Figure 24.6 Simultaneous Rewrite at the Peak of a Triangular Wave of the Slave Channel**

**Setting:**

CH0 is the master channel, which starts counting down, and CH1 represents an arbitrary slave channel. The simultaneous rewrite method B is applied.

**Description:**

(1) When TAUBnTS.TAUBnTSm = 1 is set, the value of TAUBnCDRm is copied to the TAUBnCDRm buffer.

(2) The TAUBnCDRm and TAUBnTOL registers can be written at any time.

(3) Simultaneous rewrite does not occur because it is disabled (TAUBnRSF.TAUBnRSFm = 0).

(4) The reload data trigger bit (TAUBnRDT.TAUBnRDTm) is set to 1 which sets the status flag (TAUBnRSF.TAUBnRSFm = 1), enabling simultaneous rewrite.

(5) Simultaneous rewrite does not take place at the bottom of the triangular cycle.

(6) Simultaneous rewrite takes place at the start timing of the top of the triangular cycle. The TAUBnCDRm value is loaded into the TAUBnCDRm buffer, and the TAUBnTOL.TAUBnTOLm value is loaded into the TAUBnTOL.TAUBnTOLm buffer.

(7) The counters count down and await the next simultaneous rewrite trigger. The values of TAUBnCDRm and TAUBnTOL.TAUBnTOLm can be changed again.

### 24.6.4.3 Simultaneous Rewrite when INTTAUBnIm is Generated on an Upper Channel Specified by TAUBnRDC.TAUBnRDCm (Method C1)

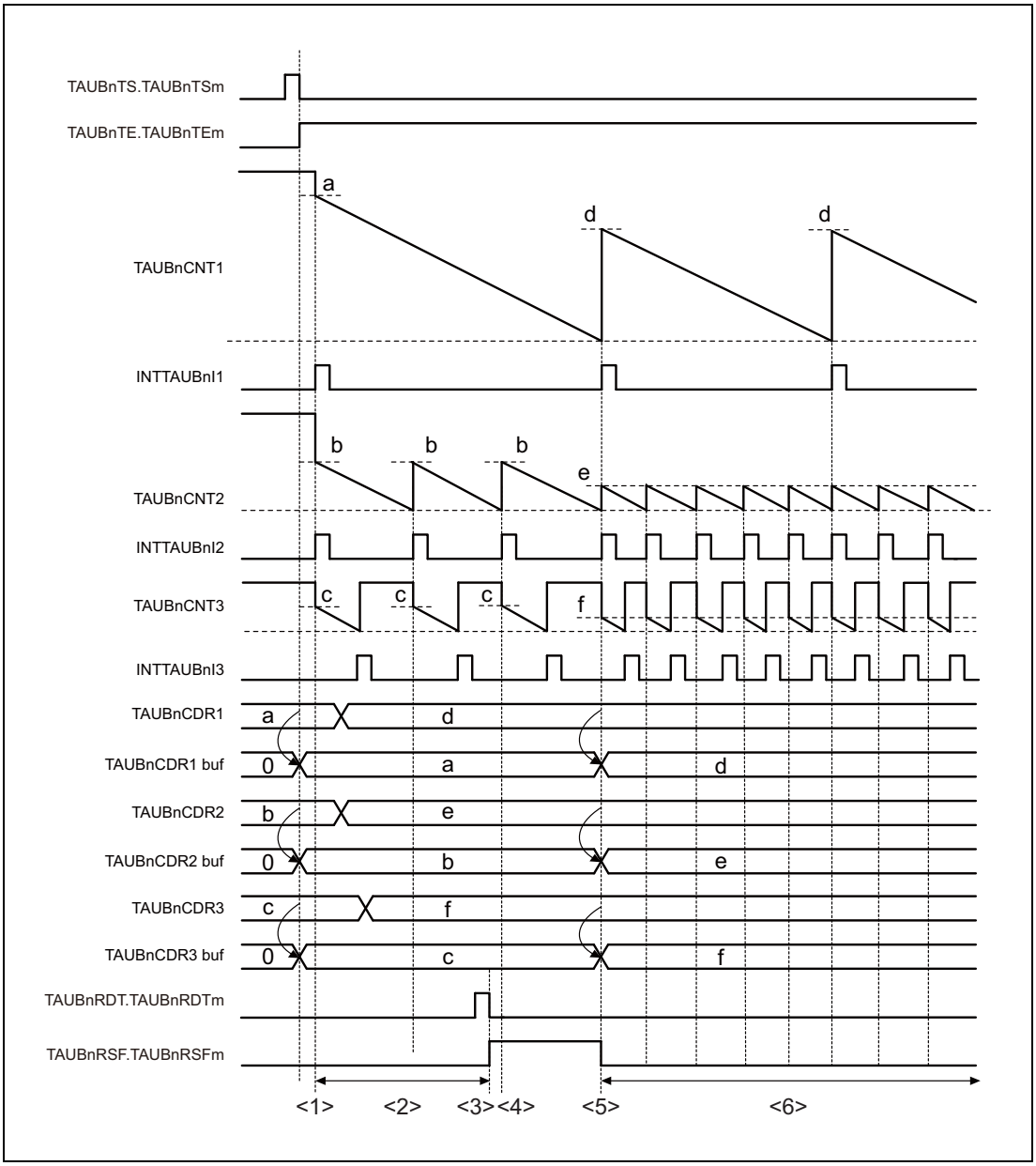



**Figure 24.7 Simultaneous Rewrite when INTTAUBnIm is Generated on an Upper Channel Specified by TAUBnRDC.TAUBnRDCm**

**Setting:**

CH1 is an upper channel used counting down, CH2 is a master channel, and CH3 is the slave channel. The simultaneous rewrite method C1 is applied. The TAUBnRDC register specifies a channel which generates simultaneous rewrite triggers.

**Description:**

(1) When TAUBnTS.TAUBnTSm is set to 1, the TAUBnCDRm value is copied to the TAUBnCDRm buffer.

(2) The TAUBnCDRm register is always ready to write.

(3) By setting the reload data trigger bit (TAUBnRDT.TAUBnRDTm) to 1, the status flag is set (TAUBnRSF.TAUBnRSFm = 1) to enable simultaneous rewrite.

(4) Simultaneous rewrite is triggered only by a CH1 interrupt. Therefore, simultaneous rewrite is not conducted even if enabled.

(5) Simultaneous rewrite is triggered by INT1 which is generated when counter 1 reaches $0000_H$. The TAUBnCDRm values are loaded into the corresponding TAUBnCDRm buffers.

(6) The counter counts down and awaits the next simultaneous rewrite trigger. The values of the TAUBnCDRm register can be rechanged.

## 24.7 Channel Output Modes

The output of the TAUBTTOUTm pin can be controlled in two ways, the latter of which can be further split into individual modes.

- By software (TAUBnTOE.TAUBnTOEm = 0)
  When controlled by software, the value written in the output register bit (TAUBnTO.TAUBnTOm) is sent to the output pin (TAUBTTOUTm).
- By TAUB signals (TAUBnTOE.TAUBnTOEm = 1)
  When controlled by TAUB signals, the output level of TAUBTTOUTm is set or reset or toggled by internal signals. The value of TAUBnTO.TAUBnTOm is updated accordingly to reflect the value of TAUBTTOUTm.
  - Independently (TAUBnTOM.TAUBnTOMm = 0)
    In case of independent operation, the output of the TAUBTTOUTm pin is only affected by settings of channel m. Therefore, independent channel operation should be selected (TAUBnTOM.TAUBnTOMm = 0).
  - Synchronously (TAUBnTOM.TAUBnTOMm = 1)
    In case of synchronous operation, the output of the TAUBTTOUTm pin is affected by settings of channel m and those of other channels. Therefore, synchronous channel operation should be selected for all synchronized channels (TAUBnTOM.TAUBnTOMm = 1).

The TAUBnTO.TAUBnTOm bit can always be read to determine the current value of TAUBTTOUTm, regardless of whether the pin is controlled by software, operated independently, or operated synchronously.

### Control bits

The settings of the control bits required to select a specific channel output mode are listed in **Table 24.38, Channel Output Modes**.

The channel output modes are described in details below.

- **Section 24.7.2, Channel Output Modes Controlled Independently by TAUBn Signals**
- **Section 24.7.3, Channel Output Modes Controlled Synchronously by TAUBn Signals**

### Batch operation of TAUBnTOm bit

Whether a set value is reflected to the TAUBnTOm bit or not is controlled by the TAUBnTOE.TAUBnTOEm bit.

The TAUBnTOm setting is written only to the bit (channel) set with TAUBnTOE.TAUBnTOEm bit = 0 when a write to the TAUBnTO register is attempted. No TAUBnTOm setting is reflected to the bit (channel) set with TAUBnTOE.TAUBnTOEm bit = 1.

**NOTE**

The TAUBnTO.TAUBnTOm bit is placed so that its bit number corresponds to a channel number.

### Output logic

Positive logic or negative logic of the output is specified by control bit TAUBnTOL.TAUBnTOLm.

The value of TAUBnTOL.TAUBnTOLm bit should be set before the counter is started. It can only be changed during operation with PWM output function or triangle PWM output function. If TAUBnTOL.TAUBnTOLm is changed after the counter starts, the output of TAUBTTOUTm is undefined.

See **Section 24.6, Simultaneous Rewrite**.

The various channel output modes and the channel output control bits are listed in **Table 24.38**.

**Table 24.38 Channel Output Modes**

| Channel Output Mode | TAUBnTOE. TAUBnTOEm | TAUBnTOM. TAUBnTOMm | TAUBnTOC. TAUBnTOCm | TAUBnTDE. TAUBnTDEm |
|---|---|---|---|---|
| **By software** | | | | |
| Independent channel output mode controlled by software | 0 | x | | |
| **By TAUB signals, independently** | | | | |
| Independent channel output mode 1 | 1 | 0 | 0 | 0 |
| Independent channel output mode 2 | | | 1 | |
| **By TAUB signals, synchronously** | | | | |
| Synchronous channel output mode 1 | 1 | 1 | 0 | 0 |
| Synchronous channel output mode 2 | | | 1 | 0 |
| Synchronous channel output mode 2 with dead time output | | | | 1 |

- All combinations not listed in this table are forbidden.
- Bits marked with an x can be set to any value.

**NOTE**

The following bits cannot be changed during count operation (TAUBnTE.TAUBnTEm = 1):

- TAUBnTOM.TAUBnTOMm
- TAUBnTOC.TAUBnTOCm
- TAUBnTDE.TAUBnTDEm

### 24.7.1 General Procedures for Specifying a Channel Output Mode

This section describes the general procedures for specifying a TAUBTTOUTm channel output mode. The prerequisite is that timer output operation is disabled (TAUBnTOE.TAUBnTOEm = 0).

(1) Set TAUBnTO.TAUBnTOm to specify the initial level of the TAUBTTOUTm output.

(2) Set channel output mode according to **Table 24.38, Channel Output Modes**, and the output logic using the TAUBnTOL.TAUBnTOLm bit.

(3) Start the counter (TAUBnTS.TAUBnTSm = 1).

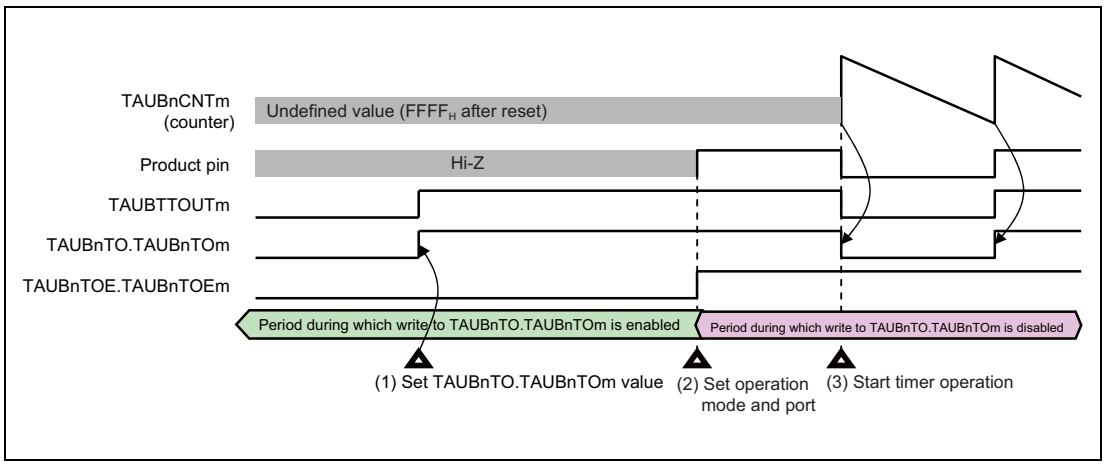



**Figure 24.8 General Procedure for Specifying a TAUBTTOUTm Channel Output Mode**

## 24.7.2 Channel Output Modes Controlled Independently by TAUBn Signals

This section lists the channel output modes that are controlled independently by TAUBn signals. The control bits used to specify a mode are listed in **Table 24.38, Channel Output Modes**.

### 24.7.2.1 Independent Channel Output Mode 1

#### Set/reset conditions

In this output mode, TAUBTTOUTm toggles when INTTAUBnIm is detected. The value of TAUBnTOL.TAUBnTOLm is ignored.

#### Prerequisites

There are no prerequisites other than those shown in **Table 24.38, Channel Output Modes**.

### 24.7.2.2 Independent Channel Output Mode 2

#### Set/reset conditions

In this output mode, TAUBTTOUTm is set when INTTAUBnIm occurs at the time of count start, and reset when INTTAUBnIm occurs due to a match between TAUBnCNTm and TAUBnCDRm.

#### Prerequisites

There are no prerequisites other than those shown in **Table 24.38, Channel Output Modes**.

## 24.7.3 Channel Output Modes Controlled Synchronously by TAUBn Signals

This section lists the channel output modes that are controlled synchronously by TAUBn signals. The control bits used to specify a mode are listed in **Table 24.38, Channel Output Modes**.

### 24.7.3.1 Synchronous Channel Output Mode 1

#### Set/reset conditions

In this output mode, INTTAUBnIm of master channel serves as a set signal and INTTAUBnIm of the slave channel as a reset signal. If INTTAUBnIm of the master channel and INTTAUBnIm of the slave channel are generated at the same time, INTTAUBnIm of the slave channel (reset signal) has priority over INTTAUBnIm (set signal) of the master channel, i.e., the master channel is ignored.

#### Prerequisites

There are no prerequisites other than those shown in **Table 24.38, Channel Output Modes**.

### 24.7.3.2 Synchronous Channel Output Mode 2

In this output mode, the operating mode should be set to count-up/-down mode. The result is trianglular wave PWM output at TAUBTTOUTm. For details, see **Section 24.14.5, Triangle PWM Output Function**.

#### Set/reset conditions

TAUBnCNTm of the slave channel counts down and up alternatively. When it passes $0001_H$ it generates an interrupt, causing TAUBTTOUTm to toggle.

**Prerequisites**

A set of two channels is required to generate the triangle PWM output. TAUBTTOUTm should be set to 0 before the function starts.

### 24.7.3.3 Synchronous Channel Output Mode 2 with Dead Time Output

In this output mode, a dead time delay is added to TAUBTTOUTm. The set/reset conditions are shown in **Figure 24.9**.

**Set/reset conditions**

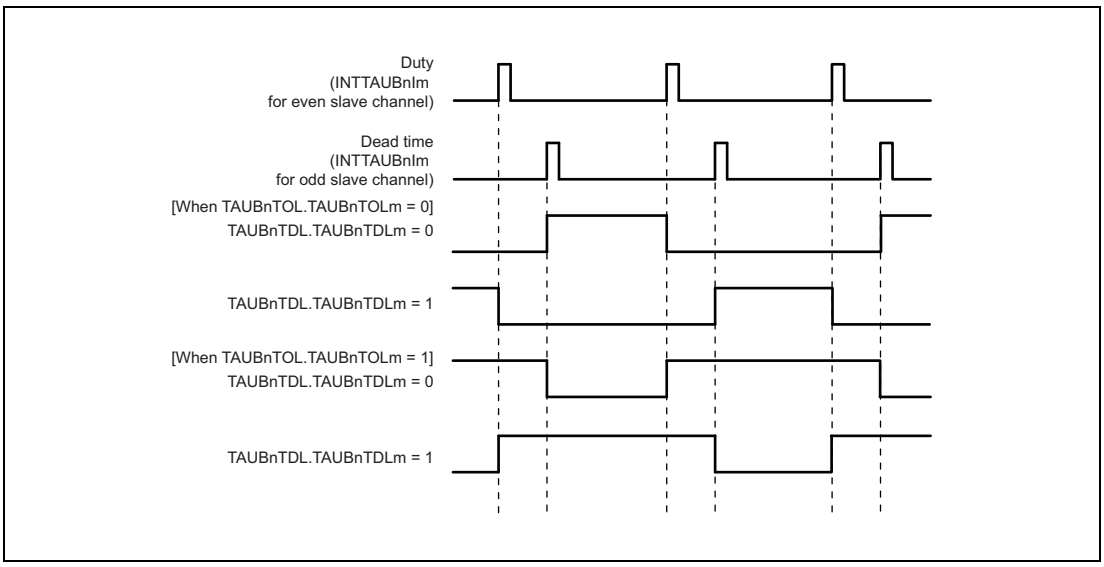



**Figure 24.9 Set/Reset Conditions for Synchronous Channel Output Mode 2 with Dead Time Output**

With regard to the edge to which dead time is added, set TAUBnTDL.TAUBnTDLm = 0 for rising edges and TAUBnTDL.TAUBnTDLm = 1 for falling edges.

**Prerequisites**

Dead time control requires a set of three channels, each operating in the following modes:

- One master channel
  The master channel should be set to interval timer mode.
- One even slave channel
  The even slave channel should be set to count-up/-down mode.
- One odd slave channel (even channel + 1)
  The odd slave channel should be set to one-count mode.

The values of the following bits should be the same for the odd channel and the even channel:

- TAUBnTOE.TAUBnTOEm
- TAUBnTOM.TAUBnTOMm
- TAUBnTOC.TAUBnTOCm
- TAUBnTDE.TAUBnTDEm

## 24.8 Start Timing in Each Operating Modes

This section describes the timing at which the counter starts after TAUBnTS.TAUBnTSm is set to 1 in each operating mode.

In all modes, the value of data register and whether or not an interrupt occurs depends on mode and register settings.

**CAUTION**

**The count start timing described in this section is for your reference. Actually, the count start timing depends on the count clock timing.**

### 24.8.1 Interval Timer Mode, Judge Mode, Capture Mode, Count-Up/-Down Mode, and Count Capture Mode

The counter starts operating at the next count clock after TAUBnTS.TAUBnTSm is set to 1. The value of data register is also loaded when the counter starts.

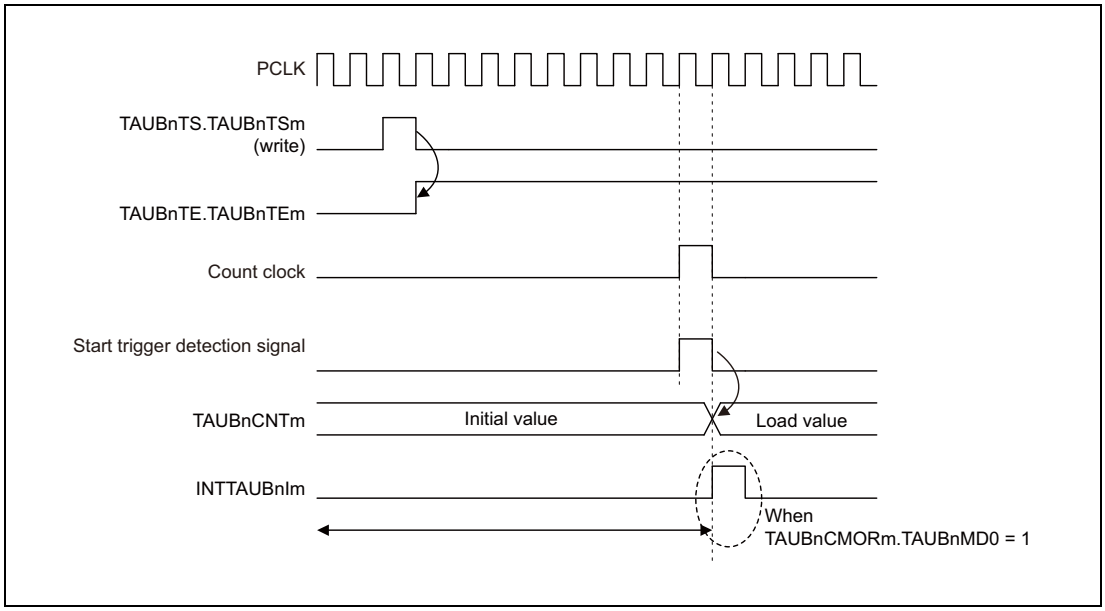



**Figure 24.10 Start Timing in Interval Timer Mode, Judge Mode, Capture Mode, Up/Down Count Mode, and Count Capture Mode**

**NOTE**

Make sure to set TAUBnCMORm.TAUBnMD0 to 0 when using the count-up/-down mode.

### 24.8.2 Event Count Mode

The value of data register is loaded as soon as TAUBnTS.TAUBnTSm is set to 1. The counter also starts immediately. The value of data register decrements when the subsequent count clock cycle starts.

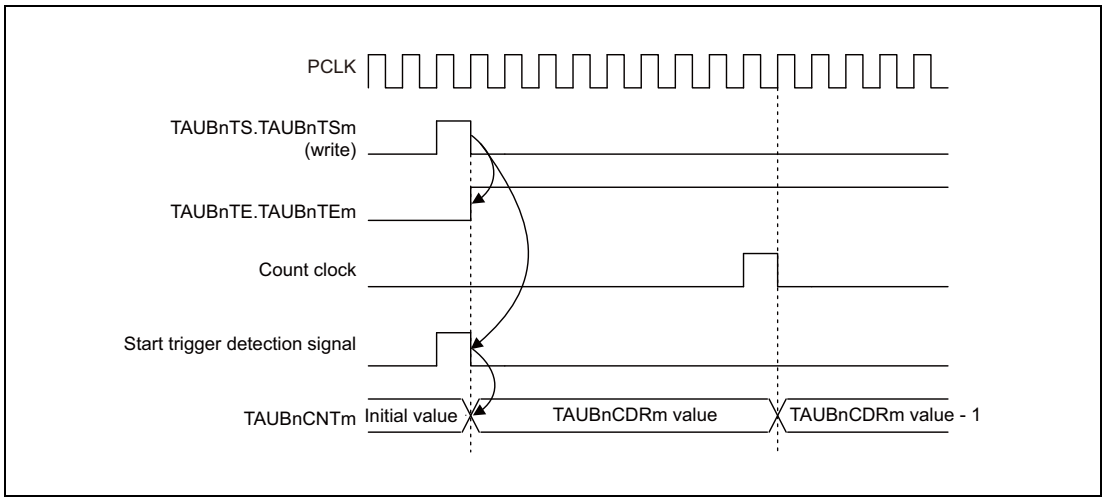



**Figure 24.11 Start Timing in Event Count Mode**

### 24.8.3 Other Operating Modes

In other operating modes, the counter operation start timing is triggered only upon detection of a valid edge of TAUBTTINm. Once the counter starts, the value of data register is also loaded. The count clock cycles, which are irrelevant to start of counter operation, determine the frequency with which all operations take place.

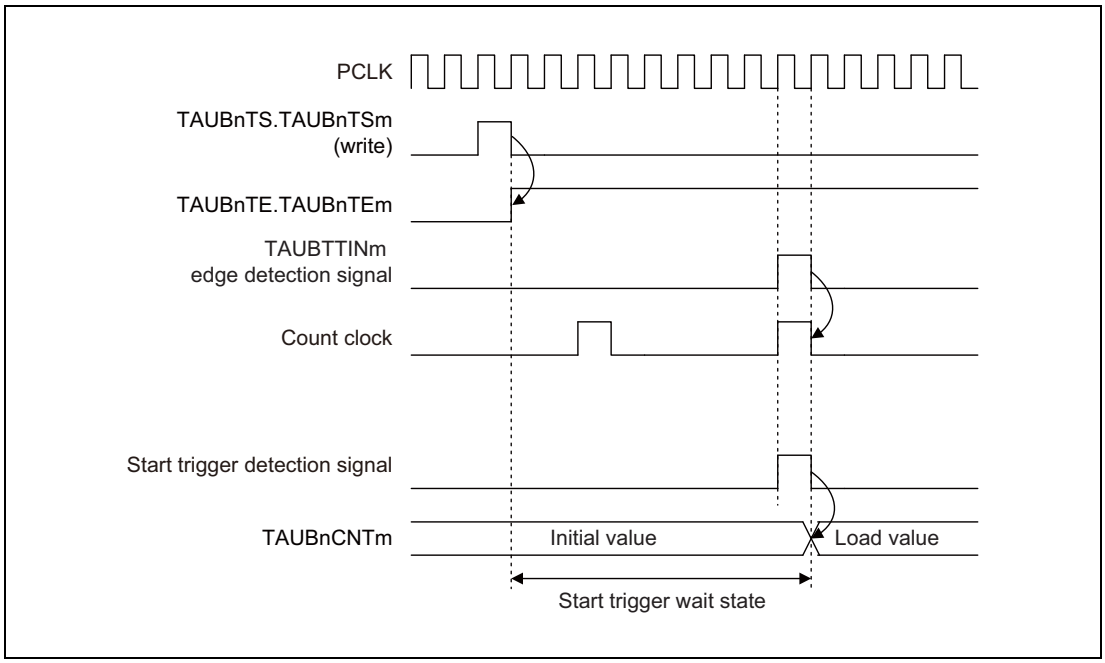



**Figure 24.12 Count Start Timing in Other Operating Modes**

## 24.9 TAUBTTOUTm Output and INTTAUBnIm Generation when Counter Starts or Restarts

When the counter starts, it is possible to specify whether an INTTAUBnIm is generated using the TAUBnCMORm.TAUBnMD0 bit. The generation of INTTAUBnIm when the TAUBnCMORm.TAUBnMD0 bit starts counting and the effect to TAUBTTOUTm depend on the selected function. For details, refer to the description of TAUBnCMORm.TAUBnMD0 of each function.

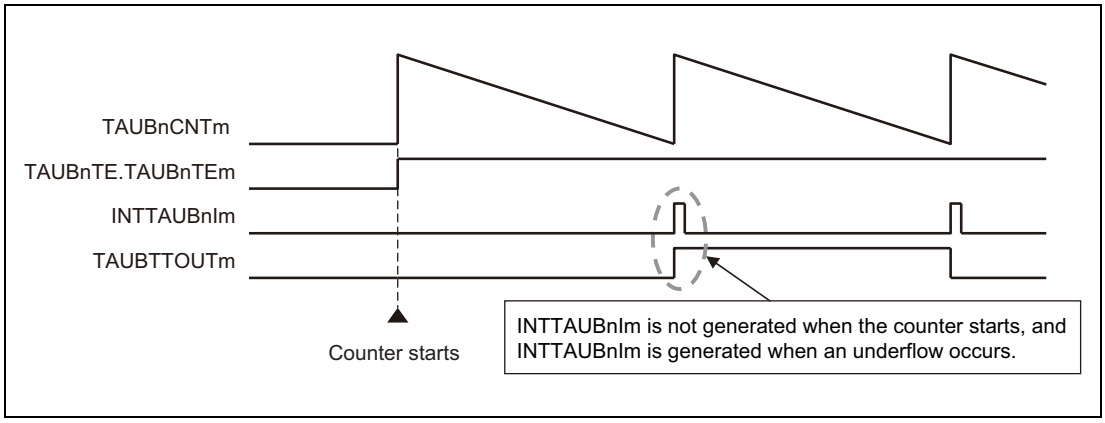



**Figure 24.13 INTTAUBnIm Generation Timing (TAUBnCMORm.TAUBnMD0=0)**

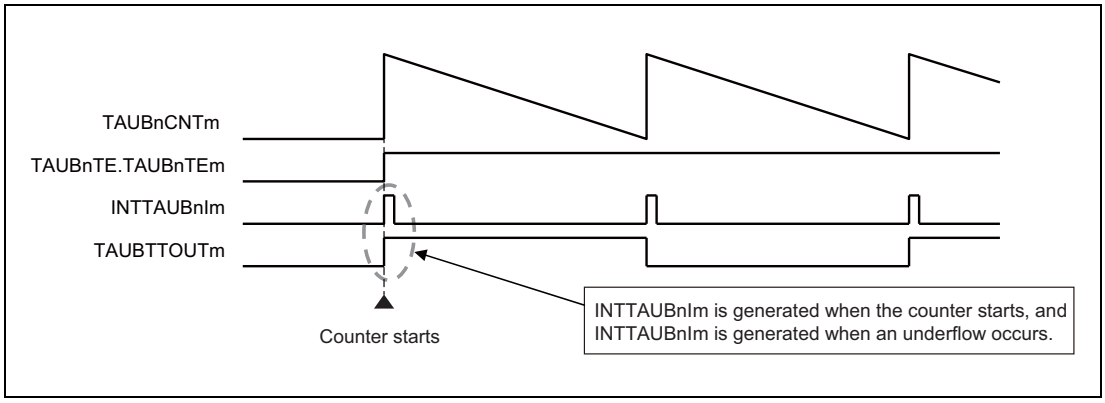



**Figure 24.14 INTTAUBnIm Generation Timing (TAUBnCMORm.TAUBnMD0=1)**

## 24.10 Interrupt Generation upon Overflow

In certain independent functions, an interrupt is not generated when the counter value reaches $FFFF_H$ and an overflow occurs during count-up. This section describes how to generate an interrupt by combining channel operation in a mode that counts up and in a mode that counts down.

The appropriate operation mode for the second channel depends on the operation mode of the first channel. Nevertheless, the principle is the same for all combinations:

- Find an operation mode for the second channel that counts down in such a manner, that it reaches $0000_H$ at the same time as the first channel overflows (TAUBnCNTm = $FFFF_H$).
- Set TAUBnCDRm of the second channel to $FFFF_H$.
- The two channels must count at the same speed (i.e. they must have the same count clock).
- Both channels are triggered by the same TAUBTTINm input.
- The trigger detection settings (TAUBnCMORm.TAUBnSTS[2:0] and TAUBnCMURm.TAUBnTIS[1:0]) must be identical for both channels.

**Result:**

The down-counter of the second channel reaches $0000_H$ at exactly the same time as the up-counter of the first channel overflows (TAUBnCNTm = $FFFF_H$). Thus the second channel generates the desired interrupt.

The following sections list the operating modes that count down that are required to match specific operating modes that count up, as well as example timing diagrams.

### 24.10.1 Example of Combination of TAUBTTINm Input Pulse Interval Measurement Function and TAUBTTINm Input Interval Timer Function

When the capture trigger is input simultaneously to TAUBTTINm of both channels, INTTAUBnIm of the TAUBTTINm input interval timer function can detect the overflow when TAUBnCNTm of the TAUBTTINm input pulse interval measurement function exceeds $FFFF_H$.

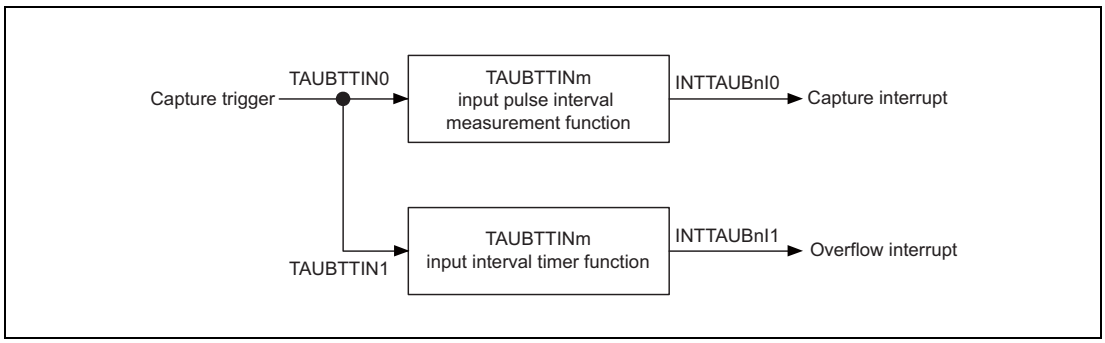



**Figure 24.15 Combination of TAUBTTINm Input Pulse Interval Measurement Function and TAUBTTINm Input Interval Timer Function**

#### Timing diagram

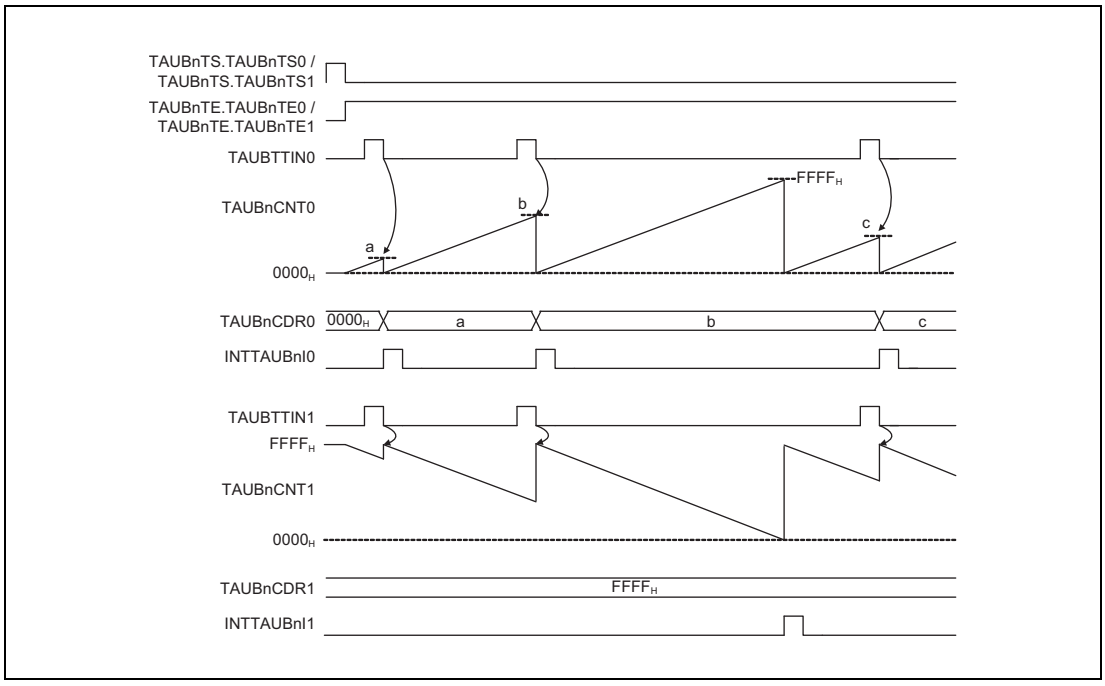



**Figure 24.16 Interrupt Generation by Combination of TAUBTTINm Input Pulse Interval Measurement Function and TAUBTTINm Input Interval Timer Function**

### 24.10.2 Example of Combination of TAUBTTINm Input Signal Width Measurement Function and Overflow Interrupt Output Function (during TAUBTTINm Width Measurement)

When the capture trigger is input simultaneously to TAUBTTINm of both channels, INTTAUBnIm of the overflow interrupt output function (during TAUBTTINm width measurement) can detect the overflow when TAUBnCNTm of the TAUBTTINm input signal width measurement function exceeds $FFFF_H$.

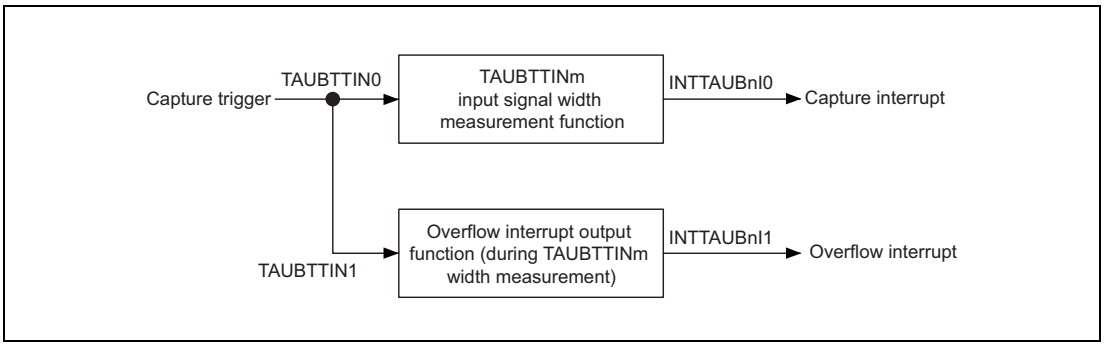



**Figure 24.17 Combination of TAUBTTINm Input Signal Width Measurement Function and Overflow Interrupt Output Function (during TAUBTTINm Width Measurement)**

**Timing diagram**

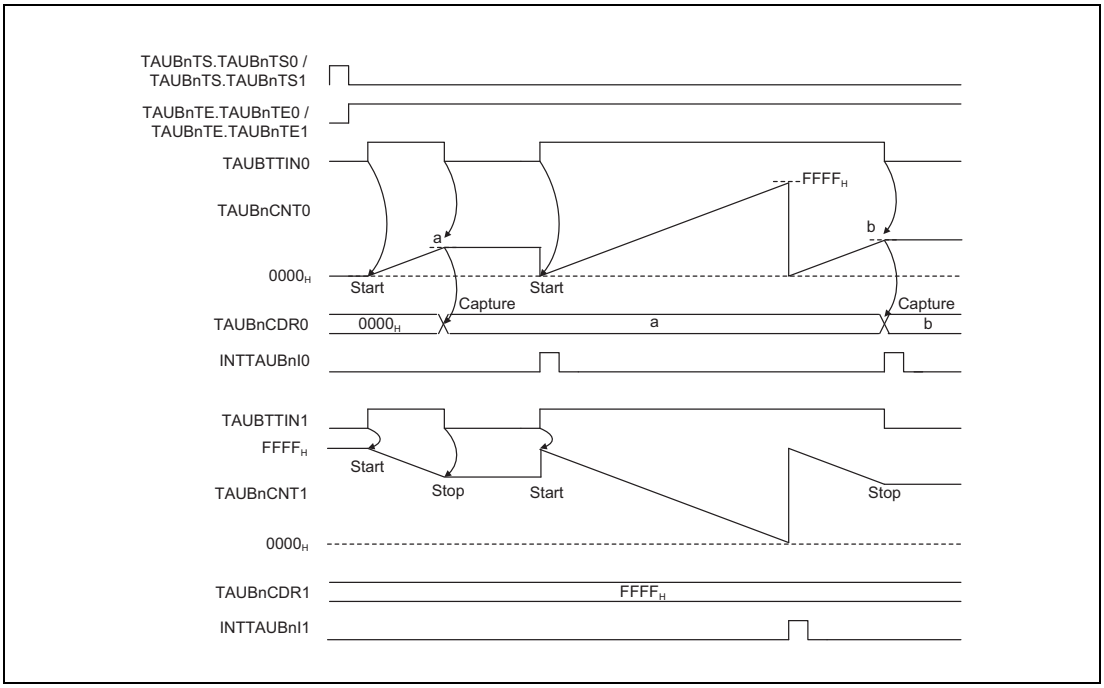



**Figure 24.18 Interrupt Generation by Combination of TAUBTTINm Input Signal Width Measurement Function and Overflow Interrupt Output Function (during TAUBTTINm Width Measurement)**

### 24.10.3 Example of Combination of TAUBTTINm Input Position Detection Function and Interval Timer Function

When the counters of both channels are enabled simultaneously, INTTAUBnIm of the interval timer function can detect the overflow when TAUBnCNTm of the TAUBTTINm input position detection function exceeds $FFFF_H$.

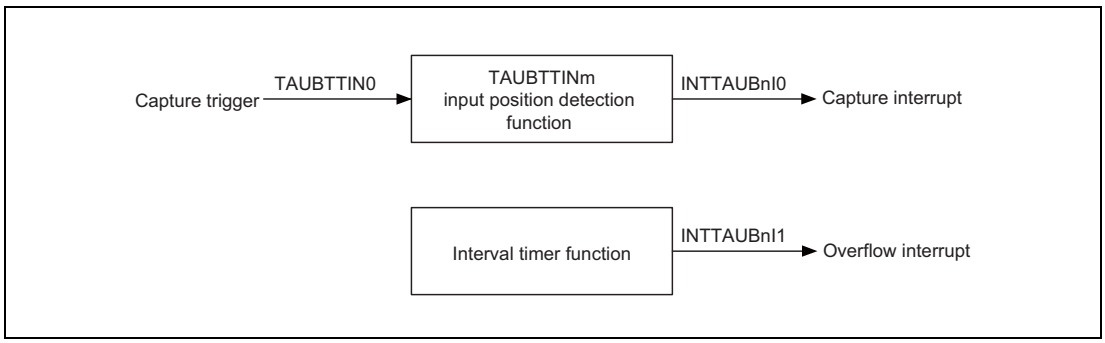



**Figure 24.19 Combination of TAUBTTINm Input Position Detection Function and Interval Timer Function**

#### Timing diagram

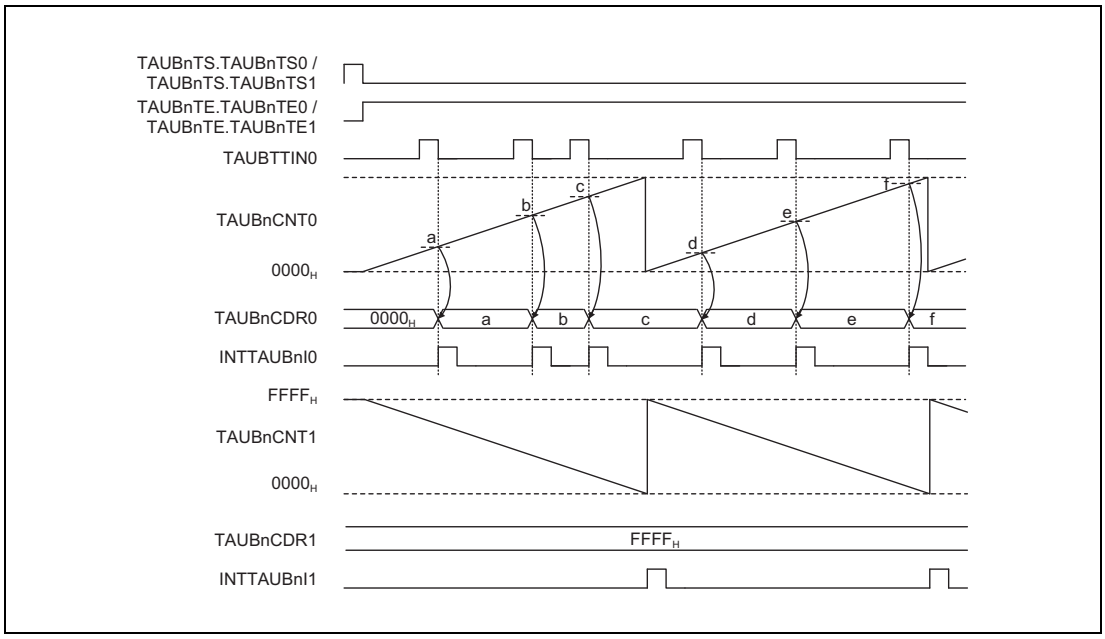



**Figure 24.20 Interrupt Generation by Combination of TAUBTTINm Input Position Detection Function and Interval Timer Function**

### 24.10.4 Example of Combination of TAUBTTINm Input Period Count Detection Function and Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection)

When the capture trigger is input simultaneously to TAUBTTINm of both channels, INTTAUBnIm of the overflow interrupt output function (during TAUBTTINm input period count detection) can detect the overflow when TAUBnCNTm of the TAUBTTINm input period count detection function exceeds $FFFF_H$.

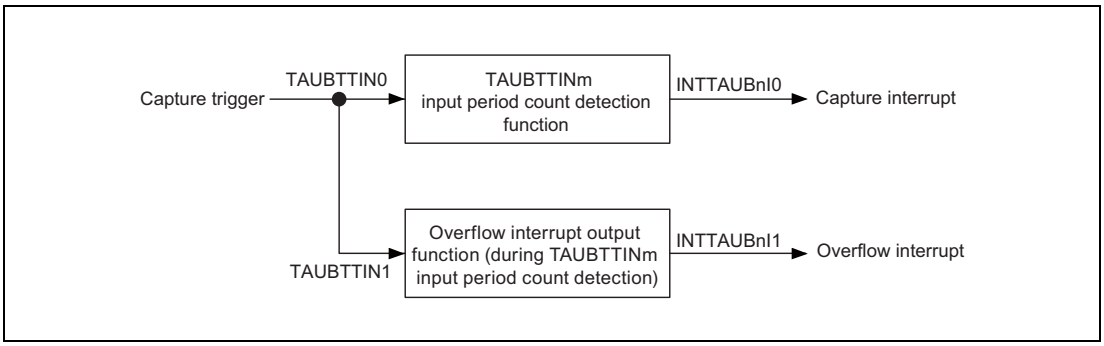



**Figure 24.21 Combination of TAUBTTINm Input Period Count Detection Function and Overflow Interrupt Output Function (TAUBTTINm Input Period Count Detection)**

**Timing diagram**

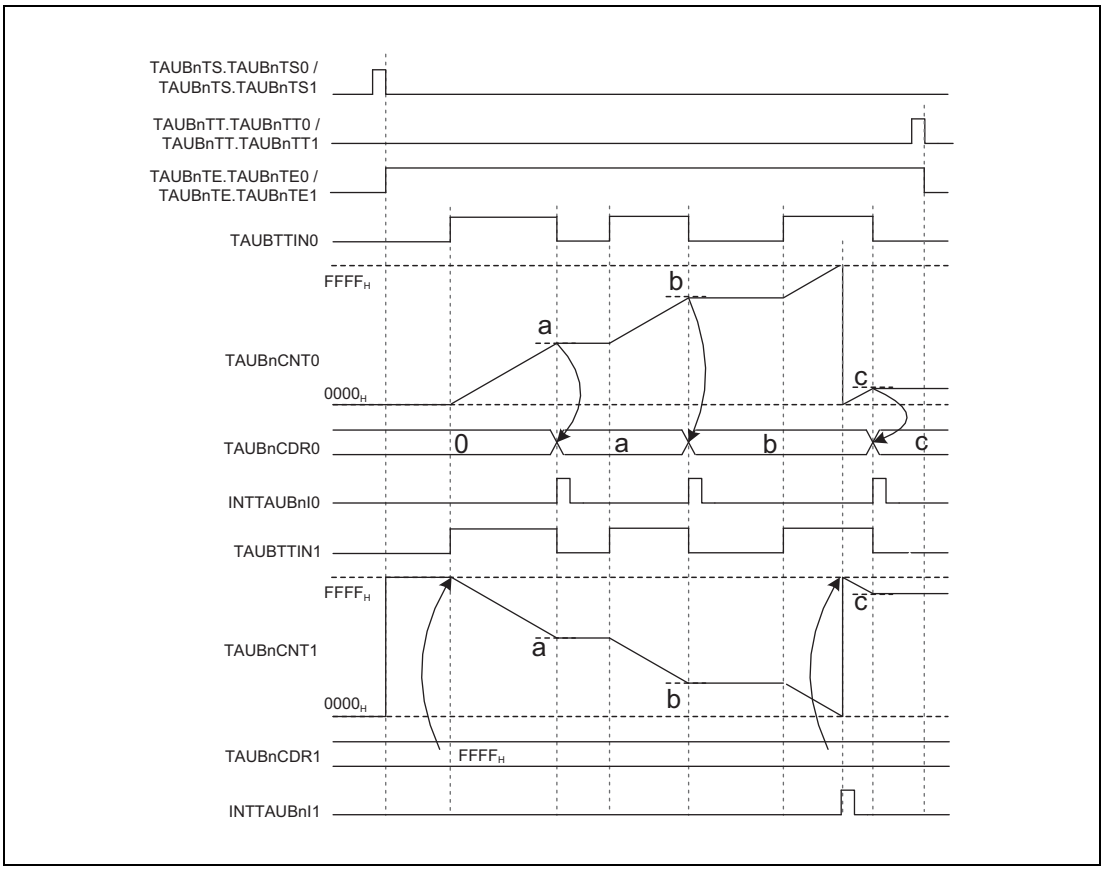



**Figure 24.22 Interrupt Generation by Combination of TAUBTTINm Input Period Count Detection Function and Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection)**

## 24.11 TAUBTTINm Edge Detection

Edge detection is based on the operation clock. This means that an edge can only be detected at the next rising edge of the operation clock. This can lead to a maximum delay of one operation clock cycle.

The following figure shows when edge detection takes place.

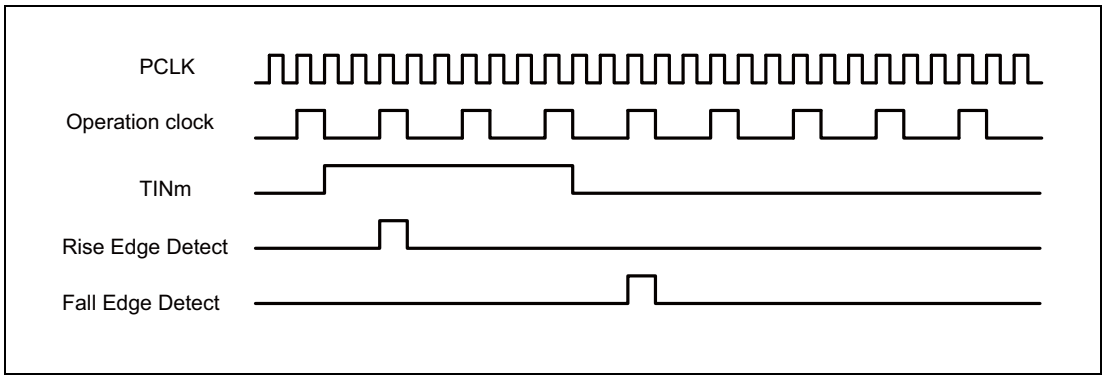



**Figure 24.23 Basic Edge Detection Timing**

**Figure 24.23** is an image of the operation timing. Actually, the delay time caused by the noise filter and the synchronization circuit between the TAUBnIm terminal and TAUBn will be generated.

# 24.12 Independent Channel Operation Functions

The following sections list the independent channel operation functions provided by the TAUB. For a general overview of independent channel operation functions, see **Section 24.2, Overview**.

## 24.12.1 Interval Timer Function

### 24.12.1.1 Overview

#### Summary

This function is used as a reference timer for generating timer interrupts (INTTAUBnIm) at regular intervals. When an interrupt is generated, the TAUBTTOUTm signal toggles, resulting in a square wave.

#### Description

The counter is enabled by setting the channel trigger bit (TAUBnTS.TAUBnTSm) to 1. This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation. The current value of TAUBnCDRm is loaded to TAUBnCNTm and the counter starts to count down from this value.

When the counter reaches $0000_H$, INTTAUBnIm is generated and the TAUBTTOUTm signal toggles. TAUBnCNTm then loads the TAUBnCDRm value and subsequently continues operation.

The value of TAUBnCDRm can be rewritten at any time, and the changed value of TAUBnCDRm is applied the next time the counter starts to count down.

The counter can be stopped by setting TAUBnTT.TAUBnTTm to 1, which in turn sets TAUBnTE.TAUBnTEm to 0. TAUBnCNTm and TAUBTTOUTm stop but retain their values. The counter can be restarted by setting TAUBnTS.TAUBnTSm to 1. The counter can also be forcibly restarted (without stopping it first) by setting TAUBnTS.TAUBnTSm to 1 during operation.

#### Conditions

If the TAUBnCMORm.TAUBnMD0 bit is set to 0, the first interrupt after a start or restart is not generated, and therefore TAUBTTOUTm does not toggle. This results in an inverted TAUBTTOUTm signal output when TAUBnCMORm.TAUBnMD0 is set to 1.

### 24.12.1.2 Equations

INTTAUBnIm cycle = count clock cycle × (TAUBnCDRm + 1)

TAUBTTOUTm square wave cycle = count clock cycle × (TAUBnCDRm + 1) × 2

### 24.12.1.3 Block Diagram and General Timing Diagram

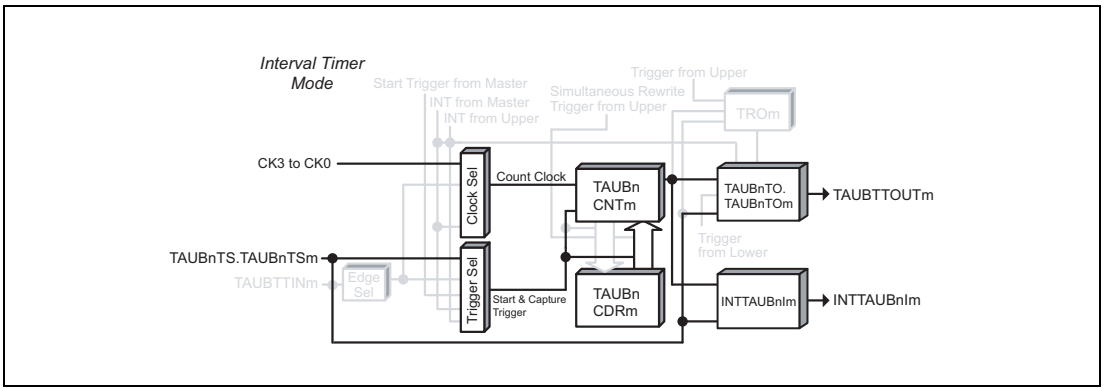



**Figure 24.24 Block Diagram for Interval Timer Function**

The following settings apply to the general timing diagram.

- INTTAUBnIm is generated at operation start (TAUBnCMORm.TAUBnMD0 = 1)

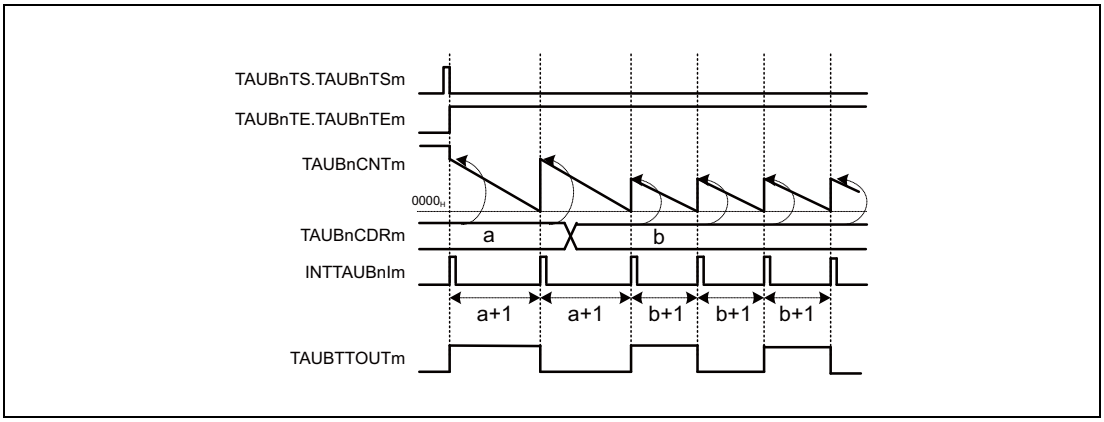



**Figure 24.25 General Timing Diagram for Interval Timer Function**

### 24.12.1.4 Register Settings

#### (1) TAUBnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.39 Contents of the TAUBnCMORm Register for Interval Timer Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $000_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0000_B$. |
| 0 | TAUBnMD0 | 0: INTTAUBnIm is not generated and TAUBTTOUTm does not toggle at operation start.<br>1: INTTAUBnIm is generated and TAUBTTOUTm toggles at operation start or restart. |

#### (2) TAUBnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.40 Contents of the TAUBnCMURm Register for Interval Timer Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Not used, so set to 00. |

### (3) Channel output mode

Table 24.41 Control Bit Settings for Independent Channel Output Mode 1

| Bit Name | Setting |
|---|---|
| TAUBnTOE.TAUBnTOEm | Write $1_B$. |
| TAUBnTOM.TAUBnTOMm | Write $0_B$. |
| TAUBnTOC.TAUBnTOCm | Write $0_B$. |
| TAUBnTOL.TAUBnTOLm | Write $0_B$. |
| TAUBnTDE.TAUBnTDEm | Write $0_B$. |
| TAUBnTDL.TAUBnTDLm | Write $0_B$. |

**NOTE**

The channel output mode can also be set to channel output mode controlled by software by setting TAUBnTOE.TAUBnTOEm = 0. TAUBTTOUTm can then be controlled independently of the interrupts.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUBnRDE, TAUBnRDS, TAUBnRDM, and TAUBnRDC) cannot be used with the Interval Timer Function. Therefore, these registers must be set to 0.

Table 24.42 Simultaneous Rewrite Settings for Interval Timer Function

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 0: Disables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: When simultaneous rewrite is disabled (TAUBnRDE.TAUBnRDEm = 0), set these bits to 0 |
| TAUBnRDM.TAUBnRDMm | |
| TAUBnRDC.TAUBnRDCm | |

### 24.12.1.5 Operating Procedure for Interval Timer Function

Table 24.43 Operating Procedure for Interval Timer Function

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Set the TAUBnCMORm and TAUBnCMURm registers as described in **Table 24.39, Contents of the TAUBnCMORm Register for Interval Timer Function** and **Table 24.40, Contents of the TAUBnCMURm Register for Interval Timer Function**<br><br>Set the value of the TAUBnCDRm register<br><br>Set the channel output mode by setting the control bits as described in **Table 24.41, Control Bit Settings for Independent Channel Output Mode 1** | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm to 1.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is set to 1 and the counter starts.<br>TAUBnCNTm loads the TAUBnCDRm value. When TAUBnCMORm.TAUBnMD0 = 1, INTTAUBnIm is generated and TAUBTTOUTm toggles. |
| During operation | The TAUBnCDRm register value can be changed at any time.<br>The TAUBnCNTm register can be read at all times. | TAUBnCNTm counts down. When the counter reaches $0000_H$:<br>• TAUBnCNTm reloads the TAUBnCDRm value and continues count operation<br>• INTTAUBnIm is generated and TAUBTTOUTm toggles. |
| Stop operation | Set TAUBnTT.TAUBnTTm to 1.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm and TAUBTTOUTm stop and retain their current values. |

Restart operation

### 24.12.1.6 Specific Timing Diagrams

#### (1) TAUBnCDRm = $0000_H$, count clock = PCLK/2

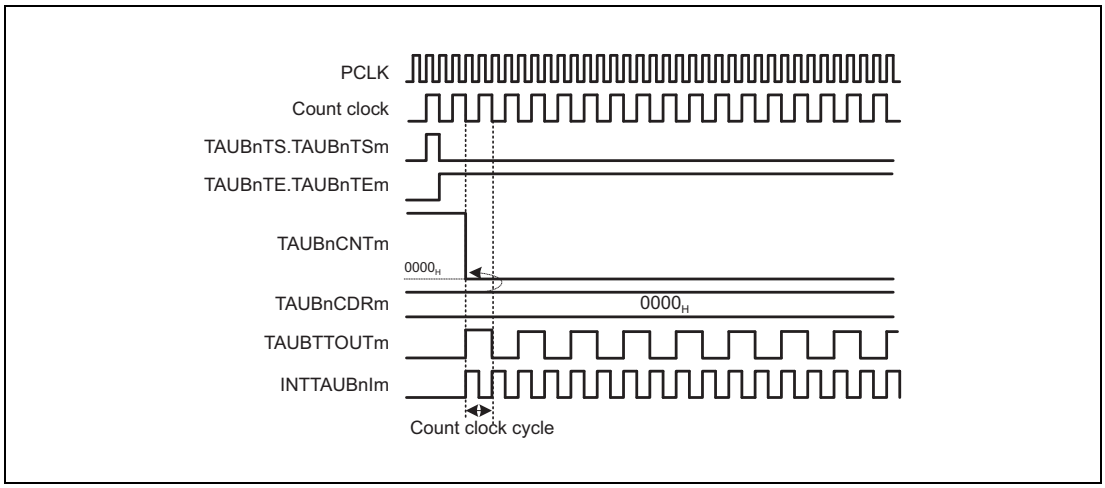



Figure 24.26 TAUBnCDRm = $0000_H$, Count Clock = PCLK/2

- TAUBnCDRm = $0000_H$, and the count clock = PCLK/2, the TAUBnCDRm value is written to TAUBnCNTm every count clock, meaning that TAUBnCNTm is always $0000_H$.
- INTTAUBnIm is generated every count clock, resulting in TAUBTTOUTm toggling every count clock.

#### (2) TAUBnCDRm = $0000_H$, count clock = PCLK

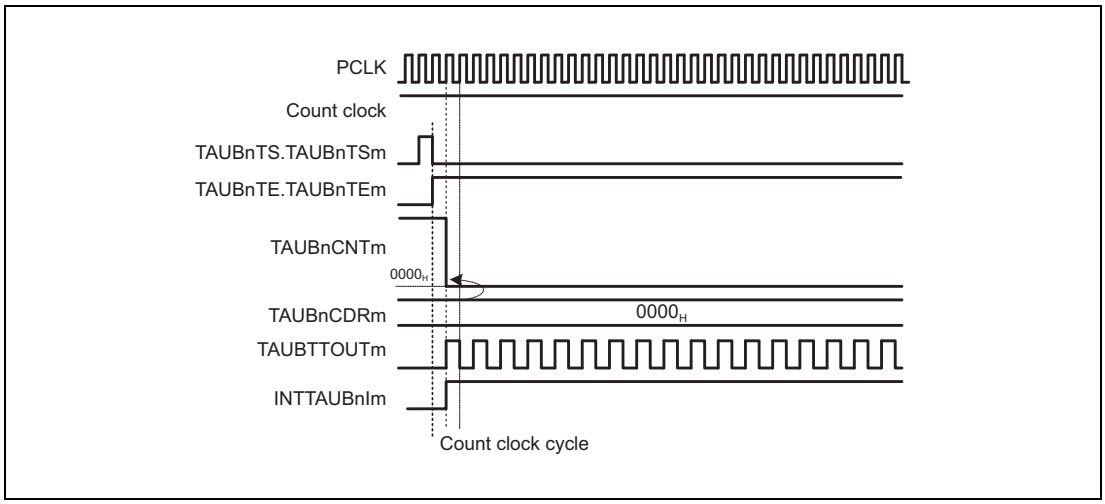



Figure 24.27 TAUBnCDRm = $0000_H$, Count Clock = PCLK

- TAUBnCDRm = $0000_H$, and the count clock = PCLK, the TAUBnCDRm value is written to TAUBnCNTm every PCLK clock, meaning that TAUBnCNTm is always $0000_H$.
- INTTAUBnIm is fixed to the high level. Though the first interrupt is generated, subsequent interrupts are not generated.
  TAUBTTOUTm is toggled every PCLK clock.

### (3) Operation stop and restart (TAUBnCMORm TAUBnMDO = 1)

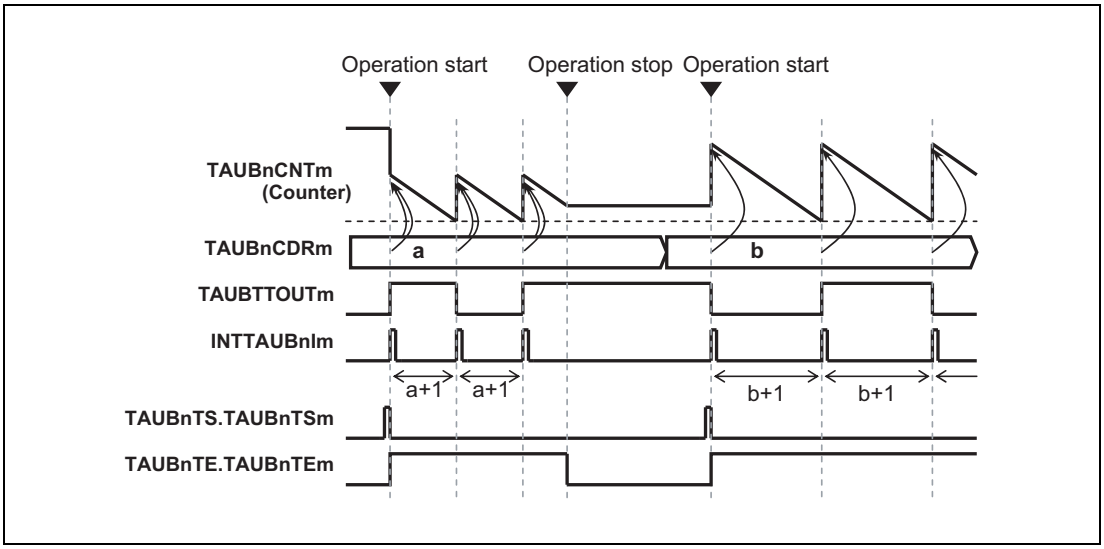



**Figure 24.28 Operation Stop and Restart, TAUBnCMORm.TAUBnMD0 = 1**

- The counter can be stopped by setting TAUBnTT.TAUBnTTm to 1, which in turn sets TAUBnTE.TAUBnTEm to 0.
- TAUBnCNTm and TAUBTTOUTm stop but retain their values.
- The counter can be restarted by setting TAUBnTS.TAUBnTSm to 1.

### (4) Operation stop and restart (TAUBnCMORm.TAUBnMD0 = 0)

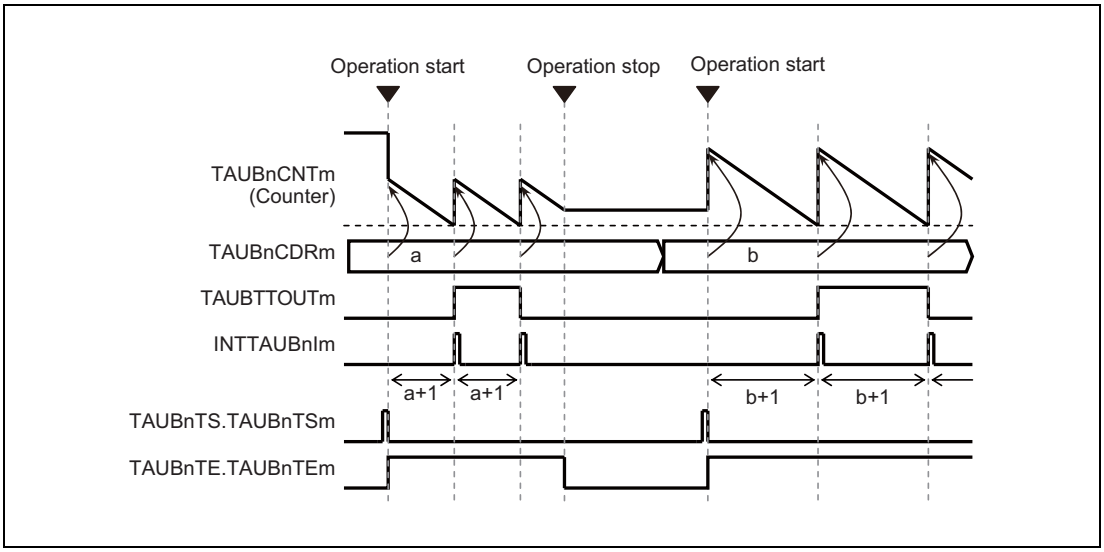



**Figure 24.29 Operation Stop and Restart, TAUBnCMORm.TAUBnMD0 = 0**

### (5) Forced restart (TAUBnCMORm.TAUBnMD0 = 1)

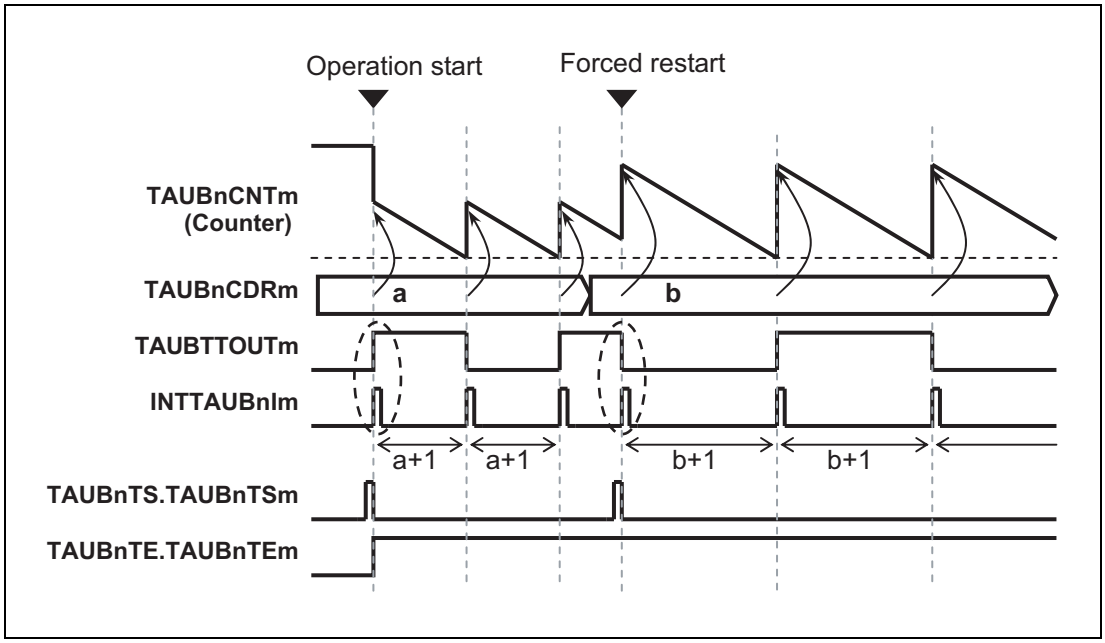



Figure 24.30 Forced Restart Operation, TAUBnCMORm.TAUBnMD0 = 1

- The counter can be forcibly restarted (without stopping it first) by setting TAUBnTS.TAUBnTSm to 1 during operation.
- If the TAUBnCMORm.TAUBnMD0 bit is set to 1, an interrupt at start or restart is generated and the output TAUBTTOUTm toggles.

### (6) Forced restart (TAUBnCMORm.TAUBnMD0 = 0)

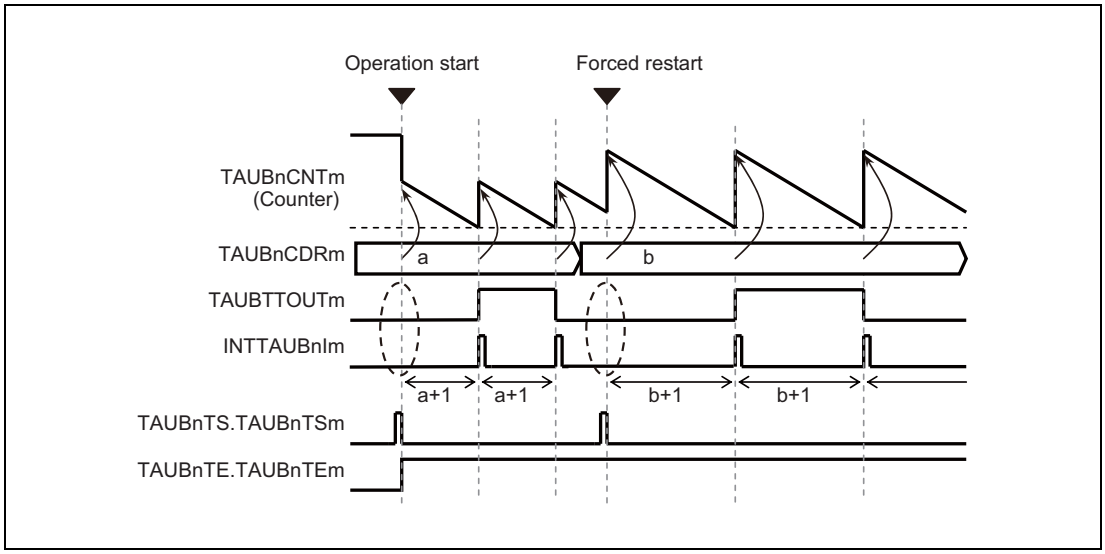



**Figure 24.31 Forced Restart Operation (TAUBnCMORm.TAUBnMD0 = 0)**

- The counter can also be forcibly restarted (without stopping it first) by setting TAUBnTS.TAUBnTSm = 1 during operation.
- If the TAUBnCMORm.TAUBnMD0 bit is set to 0, the first interrupt after a start or restart is not generated, and therefore TAUBTTOUTm does not toggle.

## 24.12.2 TAUBTTINm Input Interval Timer Function

### 24.12.2.1 Overview

#### Summary

This function is used as a reference timer for generating timer interrupts (INTTAUBnIm) at regular intervals or when a valid TAUBTTINm input edge is detected.

#### Description

The counter is enabled by setting the channel trigger bit (TAUBnTS.TAUBnTSm) to 1. This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation.

The current value of TAUBnCDRm is loaded to TAUBnCNTm and the counter starts to count down from this value.

INTTAUBnIm is generated when the counter reaches $0000_H$ or by an effective TAUBTTINm input edge. TAUBnCNTm then loads the TAUBnCDRm value and subsequently continues operation.

The value of TAUBnCDRm can be rewritten at any time, and the changed value of TAUBnCDRm is applied the next time the counter starts to count down.

The counter can be stopped by setting TAUBnTT.TAUBnTTm to 1, which in turn sets TAUBnTE.TAUBnTEm to 0. TAUBnCNTm and TAUBTTOUTm stop but retain their values. The counter can be restarted by setting TAUBnTS.TAUBnTSm to 1. The counter can also be forcibly restarted (without stopping it first) by setting TAUBnTS.TAUBnTSm to 1 during operation.

The type of edge used as the trigger is specified using the TAUBnCMURm.TAUBnTIS[1:0] bits. Either rising edge, falling edge, or rising and falling edges can be selected.

### 24.12.2.2 Block Diagram and General Timing Diagram

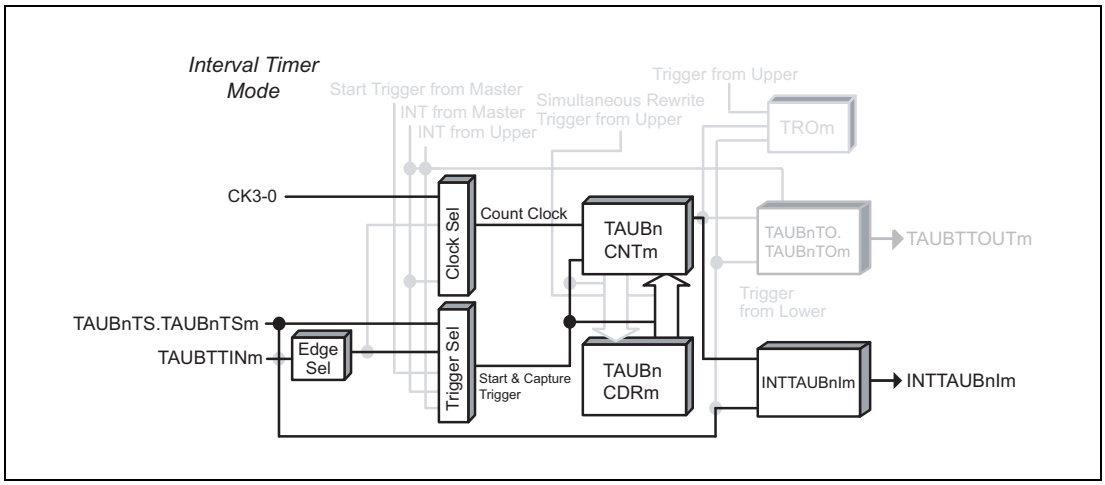



**Figure 24.32 Block Diagram for TAUBTTINm Input Interval Timer Function**

The following settings apply to the general timing diagram.

- INTTAUBnIm is generated at operation start (TAUBnCMORm.TAUBnMD0 = 1).
- Rising edge detection (TAUBnCMURm.TAUBnTIS[1:0] = $01_B$)

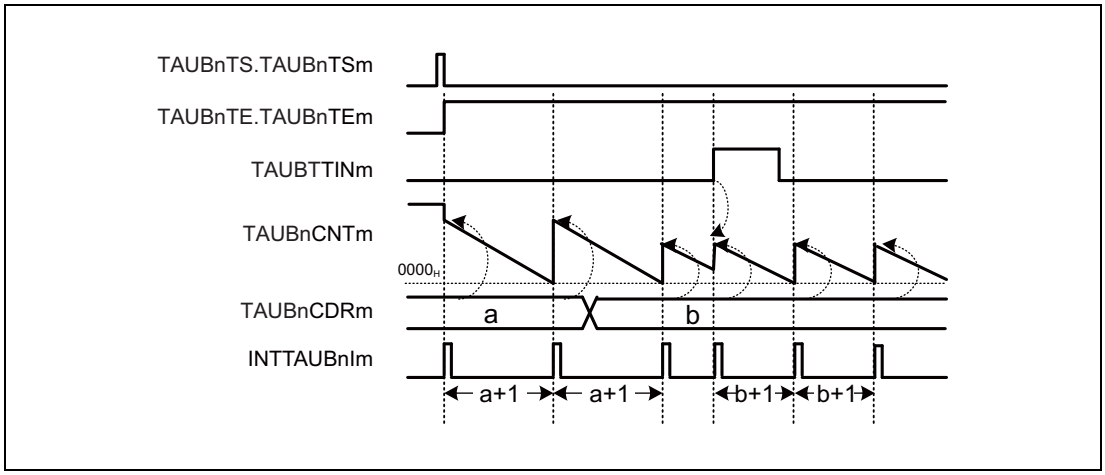



**Figure 24.33 General Timing Diagram for TAUBTTINm Input Interval Timer Function**

### 24.12.2.3 Register Settings

#### (1) TAUBnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.44 Contents of the TAUBnCMORm Register for TAUBTTINm Input Interval Timer Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $001_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0000_B$. |
| 0 | TAUBnMD0 | 0: INTTAUBnIm not generated at operation start<br>1: Generates INTTAUBnIm at operation start |

#### (2) TAUBnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.45 Contents of the TAUBnCMURm Register for TAUBTTINm Input Interval Timer Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Falling edge detection<br>01: Rising edge detection<br>10: Rising and falling edge detection<br>11: Setting prohibited |

### (3) Channel output mode

This function does not use channel output mode.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUBnRDE, TAUBnRDS, TAUBnRDM, and TAUBnRDC) cannot be used with the TAUBTTINm Input Interval Timer Function. Therefore, these registers must be set to 0.

**Table 24.46 Simultaneous Rewrite Settings for TAUBTTINm Input Interval Timer Function**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 0: Disables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: When simultaneous rewrite is disabled (TAUBnRDE.TAUBnRDEm = 0), set these bits to 0 |
| TAUBnRDM.TAUBnRDMm | |
| TAUBnRDC.TAUBnRDCm | |

### 24.12.2.4 Operating Procedure for TAUBTTINm Input Interval Timer Function

Table 24.47 Operating Procedure for TAUBTTINm Input Interval Timer Function

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Set the TAUBnCMORm and TAUBnCMURm registers as described in **Table 24.44, Contents of the TAUBnCMORm Register for TAUBTTINm Input Interval Timer Function** and **Table 24.45, Contents of the TAUBnCMURm Register for TAUBTTINm Input Interval Timer Function**<br><br>Set the value of the TAUBnCDRm register. | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm to 1.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is set to 1 and the counter starts.<br>TAUBnCNTm loads the TAUBnCDRm value.<br>When TAUBnCMORm.TAUBnMD0 = 1, INTTAUBnIm is generated. |
| During operation | The values of the TAUBnCMURm.TAUBnTIS[1:0] and the TAUBnCDRm register can be changed at any time.<br>The TAUBnCNTm register can be read at all times.<br><br>Detection of TAUBTTINm edge | TAUBnCNTm counts down. When the counter reaches $0000_H$:<br>• TAUBnCNTm reloads the TAUBnCDRm value and continues count operation<br>• INTTAUBnIm is generated<br>When a TAUBTTINm input valid edge is detected during count operation, TAUBnCNTm reloads the TAUBnCDRm value and continues count operation.<br>Afterwards, this procedure is repeated. |
| Stop operation | Set TAUBnTT.TAUBnTTm to 1<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm stop and retain their current values. |

Restart operation

### 24.12.2.5 Specific Timing Diagrams

The timing diagrams in **Section 24.12.1, Interval Timer Function** apply, and in addition the counter can also be restarted by an effective TAUBTTINm input edge.

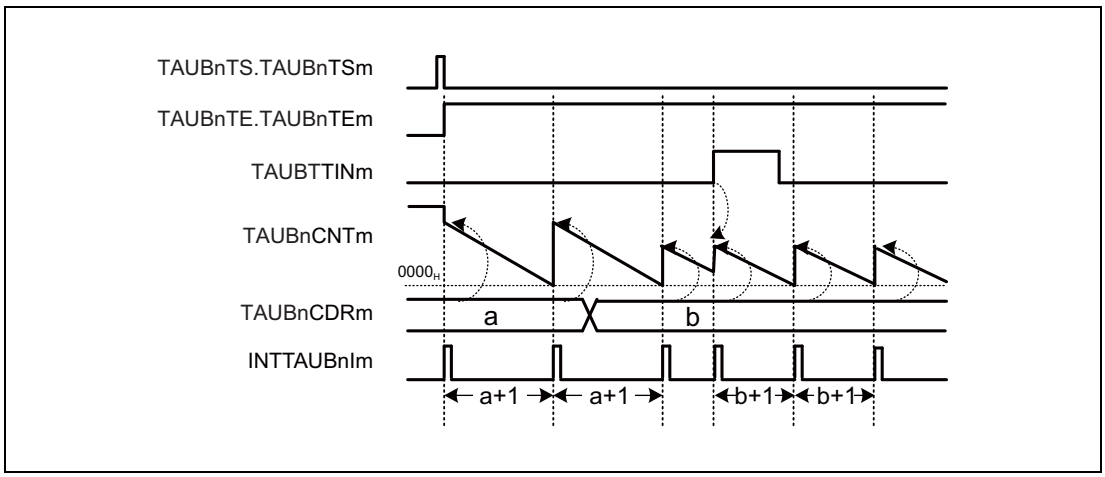



**Figure 24.34 Counter Triggered by Rising TAUBTTINm Input Edge (TAUBnCMURm.TAUBnTIS[1:0] = $01_B$), TAUBnCMORm.TAUBnMD0 = 1**

- If an effective TAUBTTINm input edge is detected, an interrupt INTTAUBnIm is generated. In this example, the effective edge is a rising edge (TAUBnCMURm.TAUBnTIS[1:0] = $01_B$).

## 24.12.3 Clock Divide Function

### 24.12.3.1 Overview

#### Summary

This function is used as a frequency divider. The frequency of the input signal TAUBTTINm is divided by a factor related to TAUBnCDRm, and an interrupt INTTAUBnIm is generated.

#### Prerequisites

- TAUBTTINm must have a fixed frequency
- The operation mode must be set to interval timer mode, see **Table 24.48, Contents of the TAUBnCMORm Register for Clock Divide Function**

#### Description

The counter is enabled by setting the channel trigger bit (TAUBnTS.TSm) to 1.
This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation. The current value of TAUBnCDRm is written to TAUBnCNTm and the counter starts to count down from this value, using TAUBTTINm as the count clock.

When the counter value reaches $0000_H$, INTTAUBnIm is generated. TAUBnCNTm then loads the TAUBnCDRm value and subsequently continues operation.

The value of TAUBnCDRm can be rewritten at any time, and the changed value of TAUBnCDRm is applied the next time the function starts to count down.

The counter can be stopped by setting TAUBnTT.TAUBnTTm = 1, which in turn sets TAUBnTE.TAUBnTEm = 0. TAUBnCNTm stops but retain their values. The function can be restarted by setting TAUBnTS.TAUBnTSm = 1. The counter can also be forcibly restarted (without stopping it first) by setting TAUBnTS.TAUBnTSm = 1 during operation.

#### Conditions

If the TAUBnCMORm.TAUBnMD0 bit is set to 0, the first interrupt after a start or restart is not generated.

**NOTE**

The TAUBTTINm input signal is sampled at the frequency of the operation clock, specified by TAUBnCMORm.TAUBnCKS[1:0] bits.

### 24.12.3.2 Block Diagram and General Timing Diagram

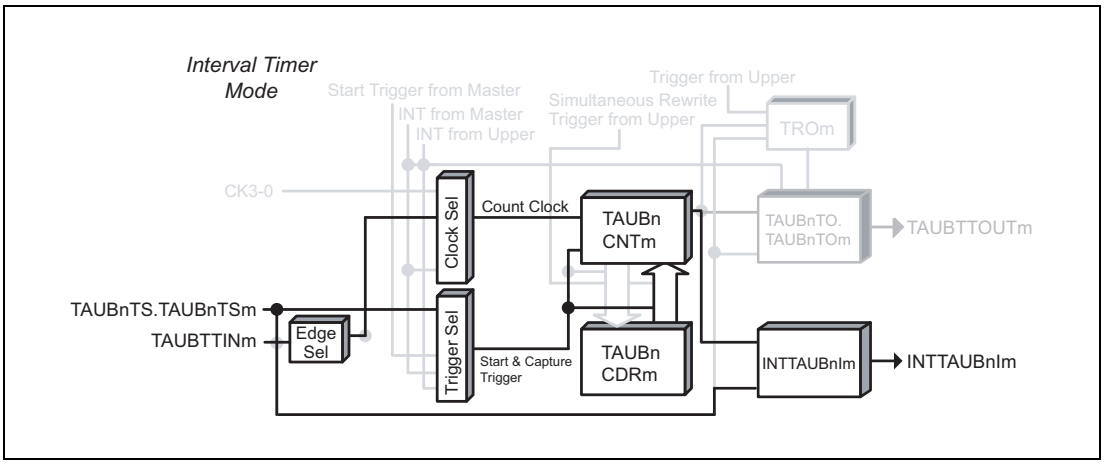



**Figure 24.35 Block Diagram for Clock Divide Function**

The following settings apply to the general timing diagram.

- INTTAUBnIm is generated at operation start (TAUBnCMORm.TAUBnMD0 = 1)
- Rising edge detection (TAUBnCMURm.TAUBnTIS[1:0] = $01_B$)

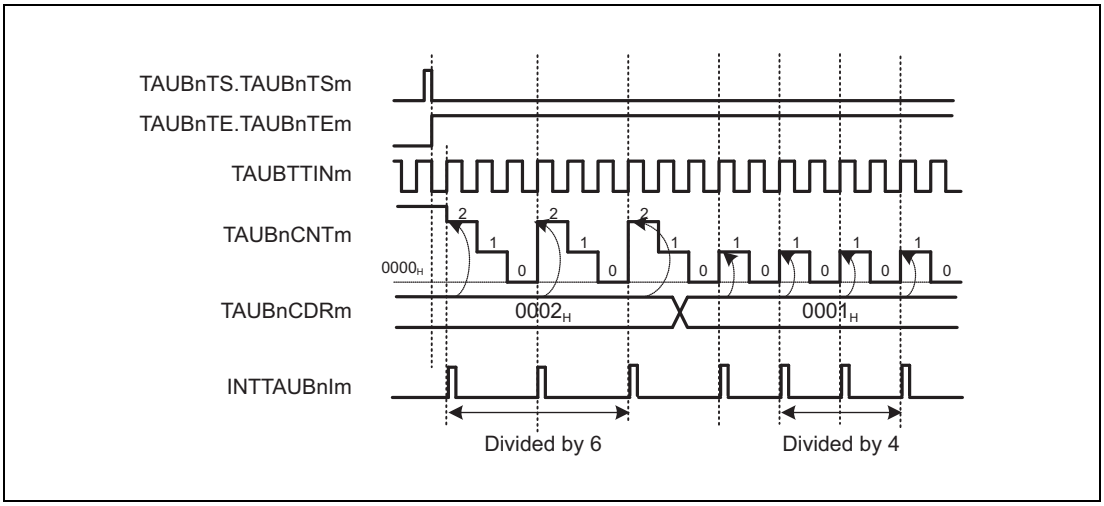



**Figure 24.36 General Timing Diagram for Clock Divide Function**

### 24.12.3.3 Register Settings

#### (1) TAUBnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.48 Contents of the TAUBnCMORm Register for Clock Divide Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $1_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $000_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0000_B$. |
| 0 | TAUBnMD0 | 0: INTTAUBnIm not generated at operation start<br>1: Generates INTTAUBnIm at operation start |

#### (2) TAUBnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.49 Contents of the TAUBnCMURm Register for Clock Divide Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Falling edge detection<br>01: Rising edge detection<br>10: Rising and falling edge detection<br>11: Setting prohibited |

### (3) Channel output mode

This function does not use channel output mode.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUBnRDE, TAUBnRDS, TAUBnRDM, and TAUBnRDC) cannot be used with the Clock Divide Function. Therefore, these registers must be set to 0.

**Table 24.50 Simultaneous Rewrite Settings for Clock Divide Function**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 0: Disables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: When simultaneous rewrite is disabled (TAUBnRDE.TAUBnRDEm = 0), set these bits to 0 |
| TAUBnRDM.TAUBnRDMm | |
| TAUBnRDC.TAUBnRDCm | |

## 24.12.3.4 Operating Procedure for Clock Divide Function

**Table 24.51 Operating Procedure for Clock Divide Function**

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Set the TAUBnCMORm and TAUBnCMURm registers as described in **Table 24.48, Contents of the TAUBnCMORm Register for Clock Divide Function** and **Table 24.49, Contents of the TAUBnCMURm Register for Clock Divide Function**<br><br>Set the value of the TAUBnCDRm register. | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm to 1.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is set to 1 and the counter starts.<br>TAUBnCNTm loads the TAUBnCDRm value.<br>When TAUBnCMORm.TAUBnMD0 is set to 1, INTTAUBnIm is generated. |
| During operation | The value of TAUBnCDRm can be changed at any time.<br>The TAUBnCNTm register can be read at all times. | When a TAUBTTINm input edge is detected, TAUBnCNTm counts down. When the counter reaches $0000_H$:<br>• TAUBnCNTm loads the TAUBnCDRm value and continues count operation<br>• INTTAUBnIm is generated |
| Stop operation | Set TAUBnTT.TAUBnTTm to 1.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm stops and retains its current value. |

Restart operation

### 24.12.3.5 Specific Timing Diagrams

#### (1) TAUBnCDRm = $0000_H$

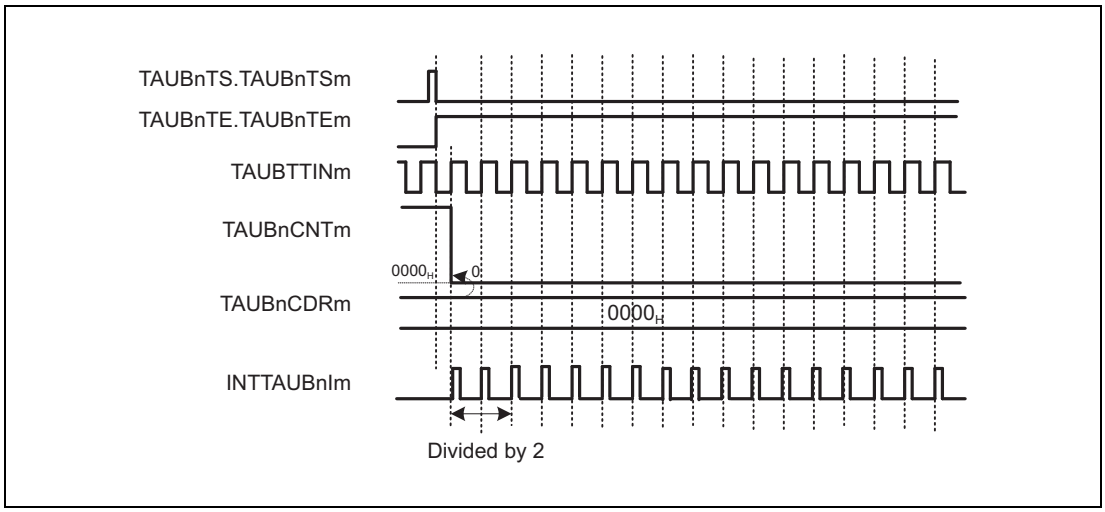



**Figure 24.37 TAUBnCDRm = $0000_H$, TAUBnCMORm.TAUBnMD0 = 1, TAUBnCMURm.TAUBnTIS[1:0] = $01_B$**

- If TAUBnCDRm is $0000_H$, TAUBnCNTm is also always $0000_H$.
- INTTAUBnIm is generated every count clock.

**Figure 24.37** is an image of the operation timing. Actually, there is a delay time caused by the noise filter and the synchronization circuit between the TAUBnIm terminal and TAUBn.

#### (2) Operation restart

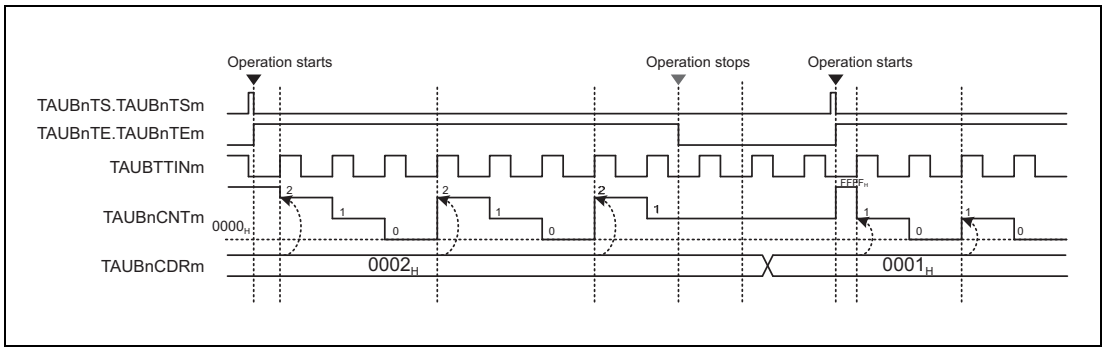



**Figure 24.38 Operation Restart (TAUBnCMORm.TAUBnMD0 = 1, TAUBnCMURm.TAUBnTIS[1:0] = $01_B$)**

### (3) Forced restart

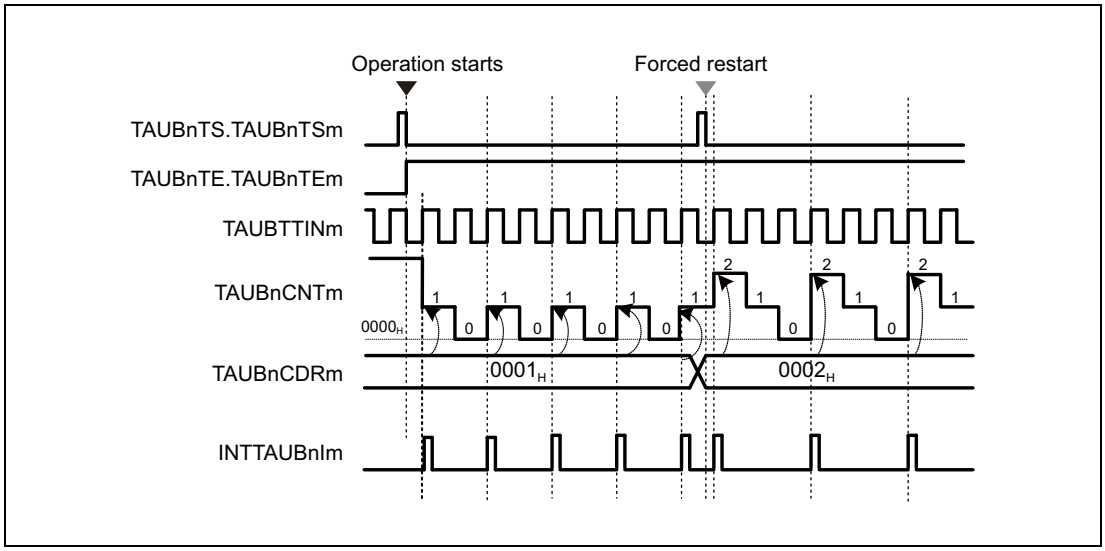



**Figure 24.39 Forced Restart (TAUBnCMORm.TAUBnMD0 = 1, TAUBnCMURm.TAUBnTIS[1:0] = $01_B$)**

To forcibly restart the counter.

- The counter can be forcibly restarted (without stopping it first) by setting TAUBnTS.TAUBnTSm = 1 during operation.
- The value of TAUBnCDRm is written to TAUBnCNTm and the count operation restarts.

## 24.12.4 External Event Count Function

### 24.12.4.1 Overview

#### Summary

This function is used as an event timer. It generates an interrupt (INTTAUBnIm) when a specific number of valid edges of TAUBTTINm input are detected.

#### Prerequisites

- The operation mode must be set to event count mode, see **Table 24.52, Contents of the TAUBnCMORm Register for External Event Count Function**
- TAUBTTOUTm is not used for this function

#### Description

The counter is enabled by setting the channel trigger bit (TAUBnTS.TAUBnTSm) to 1. This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation. When the counter starts, the current value of TAUBnCDRm is written to TAUBnCNTm.

When an effective TAUBTTINm input edge is detected, the value of TAUBnCNTm reduces by 1. TAUBnCNTm retains this value until a valid TAUBTTINm input edge is detected or the counter is restarted.

When effective edges are detected (TAUBnCDRm + 1) times, INTTAUBnIm is generated. TAUBnCNTm then loads the TAUBnCDRm value and subsequently continues to operate.

The counter can be stopped by setting TAUBnTT.TAUBnTTm to 1, which in turn sets TAUBnTE.TAUBnTEm to 0. The counter can be restarted by setting TAUBnTS.TAUBnTSm to 1. The counter can also be restarted without stopping it first (forced restart) by setting TAUBnTS.TAUBnTSm to 1 during operation.

The value of TAUBnCDRm can be rewritten at any time, and the changed value of TAUBnCDRm is applied the next time the counter starts to count down.

#### Conditions

The type of edge used as the trigger is specified by the TAUBnCMURm.TAUBnTIS[1:0] bits.

- If TAUBnCMURm.TAUBnTIS[1:0] = $00_B$, falling edges trigger the counter.
- If TAUBnCMURm.TAUBnTIS[1:0] = $01_B$, rising edges trigger the counter.
- If TAUBnCMURm.TAUBnTIS[1:0] = $10_B$, rising and falling edges trigger the counter.

### 24.12.4.2 Equations

Number of valid edges,
detected before INTTAUBnIm is generated = TAUBnCDRm + 1

### 24.12.4.3 Block Diagram and General Timing Diagram

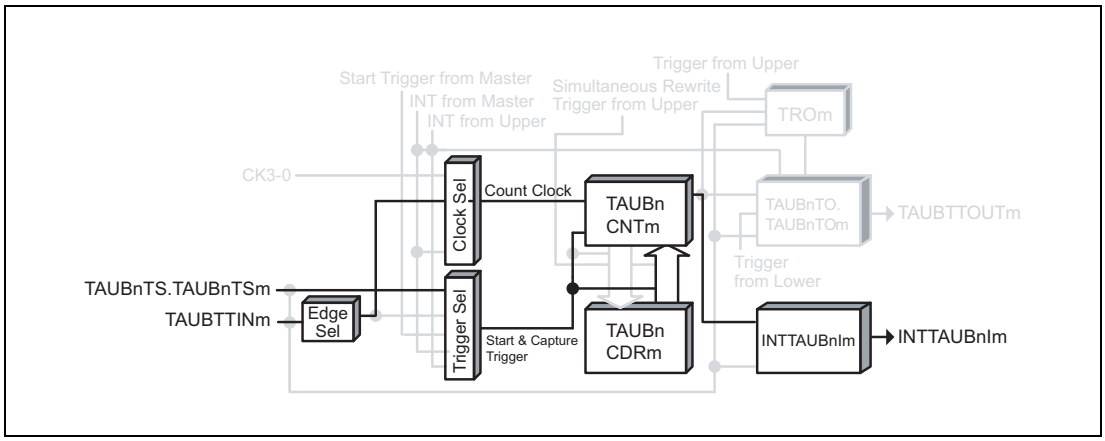



**Figure 24.40 Block Diagram for External Event Count Function**

The following settings apply to the general timing diagram.

- Rising edge detection (TAUBnCMURm.TAUBnTIS[1:0] = $01_B$)

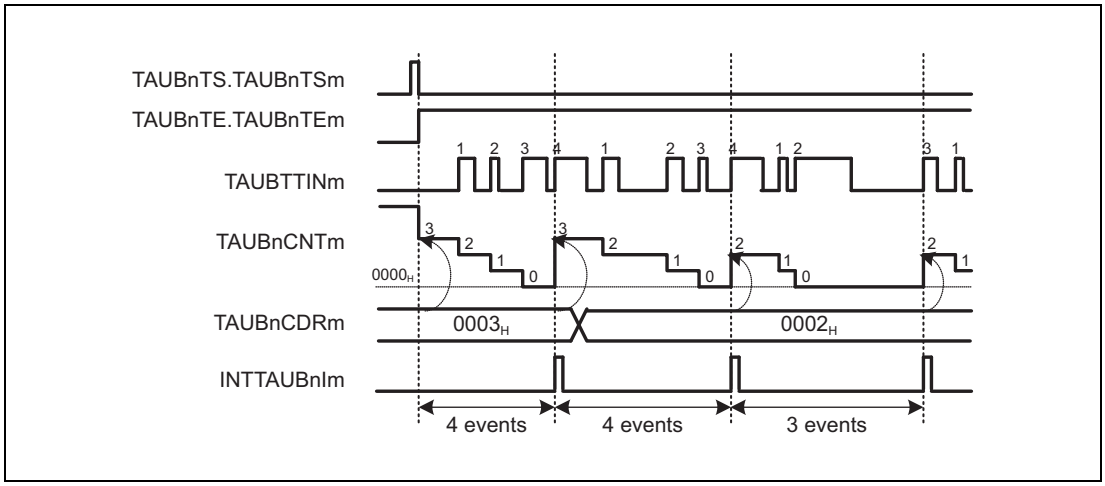



**Figure 24.41 General Timing Diagram for External Event Count Function**

### 24.12.4.4 Register Settings

#### (1) TAUBnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.52 Contents of the TAUBnCMORm Register for External Event Count Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $1_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $000_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0011_B$. |
| 0 | TAUBnMD0 | Write $0_B$. |

#### (2) TAUBnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.53 Contents of the TAUBnCMURm Register for External Event Count Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Falling edge is detected.<br>01: Rising edge is detected.<br>10: Both edges are detected.<br>11: Setting prohibited |

### (3) Channel output mode

The channel output mode is not used by this function.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUBnRDE, TAUBnRDS, TAUBnRDM, and TAUBnRDC) cannot be used with the External Event Count Function.

Therefore, these registers must be set to 0.

**Table 24.54 Simultaneous Rewrite Settings for External Event Count Function**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 0: Disables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: When simultaneous rewrite is disabled (TAUBnRDE.TAUBnRDEm = 0), set these bits to 0 |
| TAUBnRDM.TAUBnRDMm | |
| TAUBnRDC.TAUBnRDCm | |

## 24.12.4.5 Operating Procedure for External Event Count Function

**Table 24.55 Operating Procedure for External Event Count Function**

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Set the TAUBnCMORm and TAUBnCMURm registers as described in **Table 24.52, Contents of the TAUBnCMORm Register for External Event Count Function** and **Table 24.53, Contents of the TAUBnCMURm Register for External Event Count Function**<br>Set the value of the TAUBnCDRm register | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm to 1.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is set to 1 and the counter starts.<br>TAUBnCNTm loads the TAUBnCDRm value and waits for detection of the TAUBTTINm input edge. |
| During operation | Detection of TAUBTTINm edges.<br><br>The value of TAUBnCDRm can be changed at any time.<br><br>The TAUBnCNTm register can be read at any time. | TAUBnCNTm performs count-down operation each time a TAUBTTINm input edge is detected. When effective edges are detected (TAUBnCDRm + 1) times:<br>• TAUBnCNTm loads the TAUBnCDRm value and continues count operation<br>• INTTAUBnIm is generated.<br>Afterwards, this procedure is repeated. |
| Stop operation | Set TAUBnTT.TAUBnTTm to 1.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm stops and retains its current value. |

Restart operation

### 24.12.4.6 Specific Timing Diagrams

#### (1) TAUBnCDRm = $0000_H$

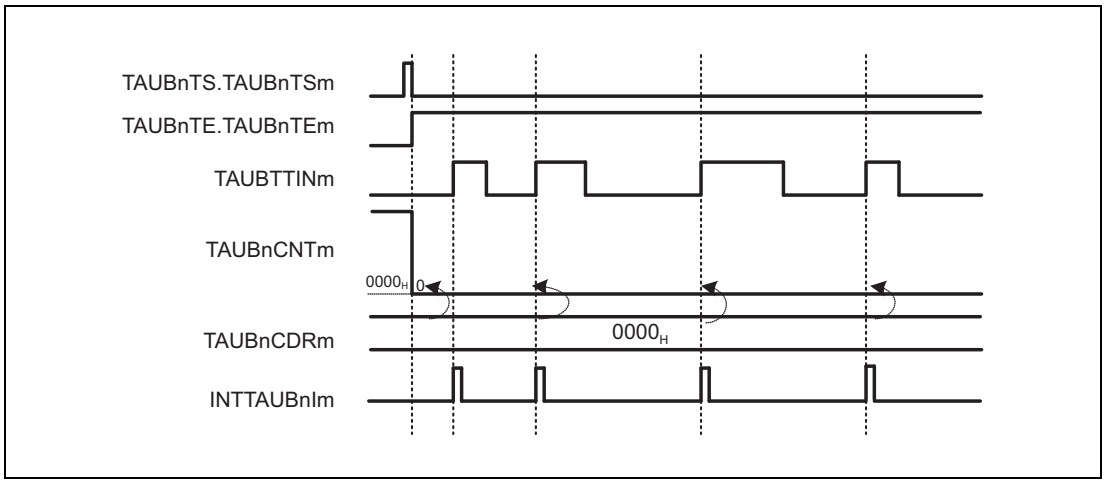



Figure 24.42 TAUBnCDRm = $0000_H$, TAUBnCMURm.TAUBnTIS[1:0] = $01_B$

- If $0000_H$ = TAUBnCDRm, $0000_H$ is loaded to TAUBnCNTm every time a valid TAUBTTINm input edge is detected.

  This means, INTTAUBnIm is generated every time a valid TAUBTTINm input edge is detected.

#### (2) Operation stop and restart

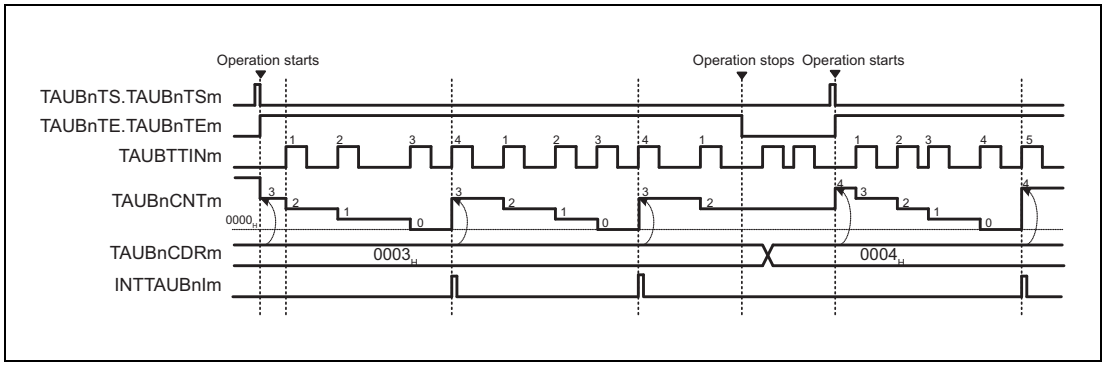



Figure 24.43 Operation Stop and Restart (TAUBnCMURm.TAUBnTIS[1:0] = $01_B$)

- The counter can be stopped by setting TAUBnTT.TAUBnTTm to 1, which in turn sets TAUBnTE.TAUBnTEm to 0.
- TAUBnCNTm stops and the current value is retained. TAUBTTINm continues and TAUBnCNTm ignores the valid edge.
- The counter can be restarted by setting TAUBnTS.TAUBnTSm to 1. TAUBnCNTm loads the TAUBnCDRm value and restarts count operation.

### (3) Forced restart

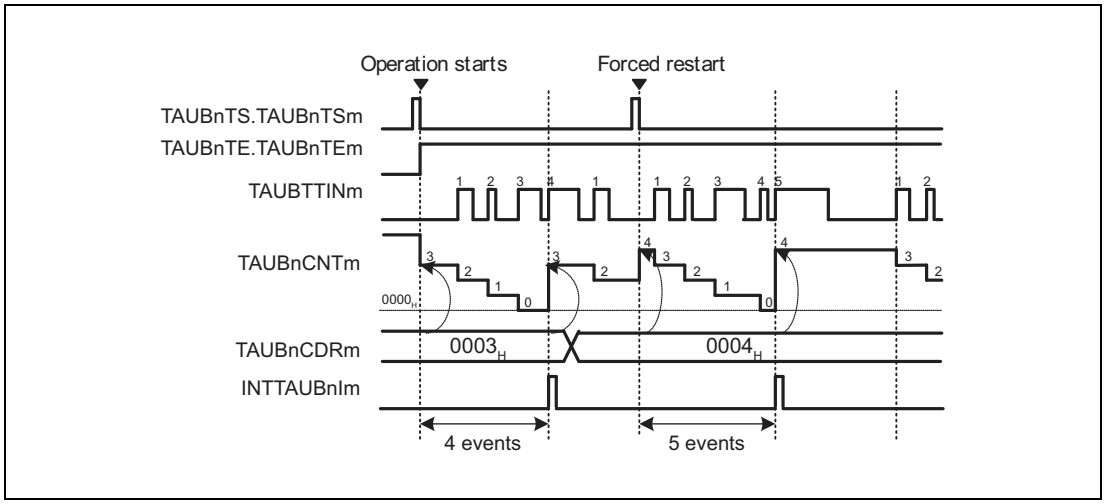



**Figure 24.44 Forced Restart (TAUBnCMURm.TAUBnTIS[1:0] = $01_B$)**

A forced restart applies the new TAUBnCDRm value to TAUBnCNTm immediately.

- The counter can be restarted (without stopping it first), by setting TAUBnTS.TAUBnTSm to 1 during operation.
- The value of TAUBnCDRm is loaded to TAUBnCNTm and the counter awaits the next valid TAUBTTINm input edge.

## 24.12.5 One-Pulse Output Function

### 24.12.5.1 Overview

#### Summary

This function generates an interrupt (INTTAUBnIm) when a valid TAUBTTINm input edge is detected and subsequently, in a specific interval. TAUBTTINm input signal pulses that occur within the defined interval are ignored.

#### Prerequisites

- The operation mode must be set to pulse one-count mode. (See **Table 24.56, Contents of the TAUBnCMORm Register for One-Pulse Output Function**).
- Trigger detection must be disabled during counting (TAUBnCMORm.TAUBnMD0 = 0).

#### Description

The counter is enabled by setting the channel trigger bit (TAUBnTS.TAUBnTSm) to 1.
This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation.

The counter starts when a valid TAUBTTINm input edge is detected. The value of TAUBnCDRm is written to TAUBnCNTm and the counter starts to count down from the TAUBnCDRm value, and an interrupt is generated.

When the counter reaches $0001_H$ an interrupt is generated. The counter stops at $0000_H$ and awaits the next effective TAUBTTINm input edge.

When the counter is counting down, further TAUBTTINm input signals are ignored, i.e. the counter does not reset.

The value of TAUBnCDRm can be rewritten at any time, and the changed value of TAUBnCDRm is applied the next time the counter starts to count down.

#### Conditions

The type of edge used as the trigger is specified by the TAUBnCMURm.TIS[1:0] bits.

- If TAUBnCMURm.TAUBnTIS[1:0] = $00_B$, falling edges trigger the counter.
- If TAUBnCMURm.TAUBnTIS[1:0] = $01_B$, rising edges trigger the counter.
- If TAUBnCMURm.TAUBnTIS[1:0] = $10_B$, rising and falling edges trigger the counter.

### 24.12.5.2 Equations

Interval between TAUBTTINm and INTTAUBnIm = count clock cycle × TAUBnCDRm

### 24.12.5.3 Block Diagram and General Timing Diagram

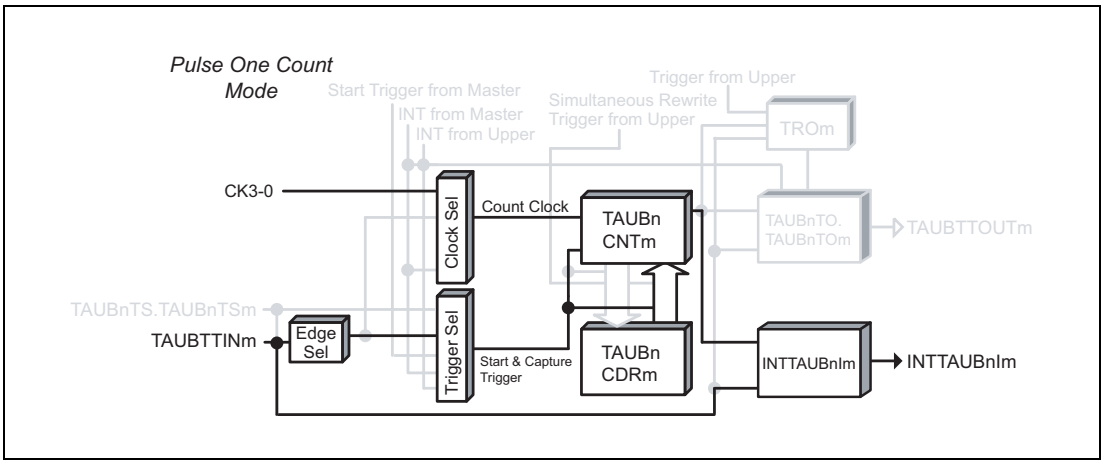



**Figure 24.45 Block Diagram for One-Pulse Output Function**

The following settings apply to the general timing diagram.

- Falling edge detection (TAUBnCMURm.TAUBnTIS[1:0] = $00_B$)

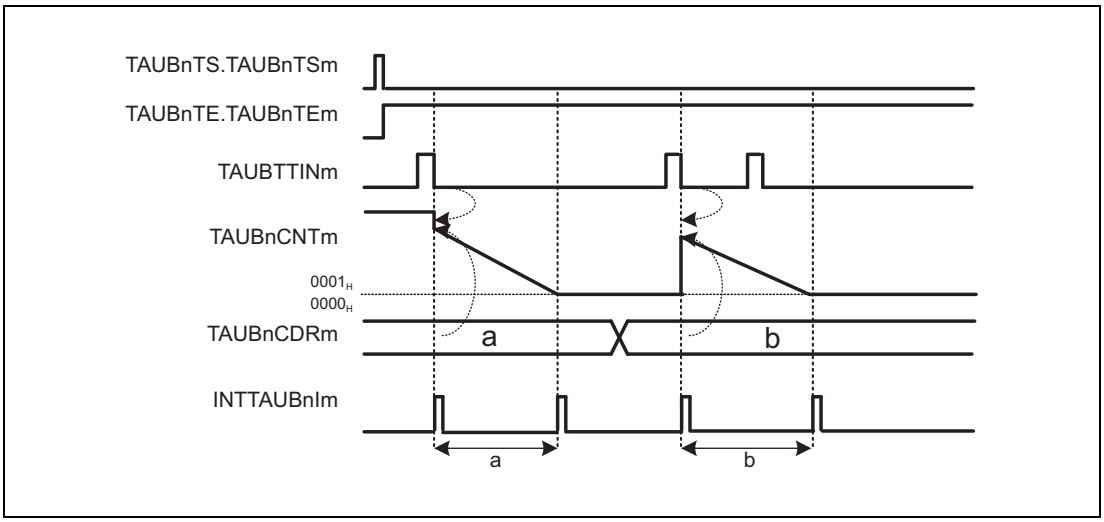



**Figure 24.46 General Timing Diagram for One-Pulse Output Function**

### 24.12.5.4 Register Settings

#### (1) TAUBnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.56 Contents of the TAUBnCMORm Register for One-Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $001_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $1010_B$. |
| 0 | TAUBnMD0 | Write $0_B$. |

#### (2) TAUBnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.57 Contents of the TAUBnCMURm Register for One-Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Falling edge detection<br>01: Rising edge detection<br>10: Rising and falling edge detection<br>11: Setting prohibited |

### (3) Channel output mode

This function does not use channel output mode.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUBnRDE, TAUBnRDS, TAUBnRDM, and TAUBnRDC) cannot be used with the One-Pulse Output Function.
Therefore, these registers must be set to 0.

**Table 24.58 Simultaneous Rewrite Settings for One-Pulse Output Function**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 0: Disables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: When simultaneous rewrite is disabled (TAUBnRDE.TAUBnRDEm = 0), set these bits to 0 |
| TAUBnRDM.TAUBnRDMm | |
| TAUBnRDC.TAUBnRDCm | |

## 24.12.5.5 Operating Procedure for One-Pulse Output Function

**Table 24.59 Operating Procedure for One-Pulse Output Function**

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Set the TAUBnCMORm and TAUBnCMURm registers as described in**Table 24.56, Contents of the TAUBnCMORm Register for One-Pulse Output Function** and **Table 24.57, Contents of the TAUBnCMURm Register for One-Pulse Output Function**<br><br>Set the value of the TAUBnCDRm register | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm to 1.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0.<br><br>Detection of TAUBTTINm start edge | TAUBnTE.TAUBnTEm is set to 1 and TAUBnCNTm waits for detection of the TAUBTTINm start edge.<br><br>When a start edge is detected, TAUBnCNTm loads the TAUBnCDRm value. |
| During operation | The value of TAUBnCDRm can be changed at any time.<br>The TAUBnCNTm register can be read at all times. | INTTAUBnIm is generated when TAUBnCNTm starts.<br><br>TAUBnCNTm counts down. When the counter reaches $0001_H$, INTTAUBnIm is generated.<br><br>TAUBnCNTm stops counting and waits for a trigger.<br><br>If a trigger occurs while TAUBnCNTm is counting, the trigger is ignored.<br><br>Afterwards, this procedure is repeated. |
| Stop operation | Set TAUBnTT.TAUBnTTm to 1.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm stops and retains its current value. |

Restart operation

## 24.12.6 TAUBTTINm Input Pulse Interval Measurement Function

### 24.12.6.1 Overview

#### Summary

This function captures the count value and uses this value and the overflow bit TAUBnCSRm.TAUBnOVF to measure the interval of the TAUBTTINm input signal.

#### Prerequisites

TAUBTTOUTm is not used for this function

#### Description

The counter is enabled by setting the channel trigger bit (TAUBnTS.TAUBnTSm) to 1.
This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation. The counter TAUBnCNTm starts counting up from $0000_H$. When a valid TAUBTTINm edge is detected, the value of TAUBnCNTm is captured, transferred to TAUBnCDRm, and an interrupt INTTAUBnIm is generated. The counter resets to $0000_H$ and subsequently continues operation.

If the counter reaches $FFFF_H$ before a valid TAUBTTINm edge is detected, it overflows to $0000_H$. The counter is reset to $0000_H$ and subsequently continues operation. The values transferred to TAUBnCDRm and TAUBnCSRm.TAUBnOVF respectively depend on the values of bits TAUBnCMORm.TAUBnCOS[1:0].

**Table 24.60 Effects of an Overflow**

| TAUBnCMORm. TAUBnCOS[1:0] | When Overflow Occurs | | When a Valid TAUBTTINm Input is then Detected | |
|---|---|---|---|---|
| | TAUBnCDRm | TAUBnCSRm. TAUBnOVF | TAUBnCDRm, TAUBnCNTm | TAUBnCSRm. TAUBnOVF |
| 00 | Unchanged | 0 | TAUBnCNTm loaded to TAUBnCDRm | 1 |
| 01 | | 1 | | |
| 10 | Set to $FFFF_H$ | 0 | TAUBnCNTm set to 0, TAUBnCDRm unchanged | Unchanged |
| 11 | | 1 | | |

If TAUBnCMORm.TAUBnCOS[0] is 1, the overflow bit TAUBnCSRm.TAUBnOVF can only be cleared by setting TAUBnCSCm.TAUBnCLOV = 1.

The combination of the value of TAUBnCDRm and TAUBnCSRm.TAUBnOVF can be used to deduce the interval of the TAUBTTINm signal. However, if an overflow occurs multiple times before a valid TAUBTTINm input is detected, the overflow bit TAUBnCSRm.TAUBnOVF cannot indicate this.

The function can be stopped by setting TAUBnTT.TAUBnTTm = 1, which in turn sets TAUBnTE.TAUBnTEm = 0. TAUBnCNTm stops but retains its value. While the function is stopped, TAUBTTINm input valid edge detection and TAUBnCNTm capture are not performed.

The counter is reset to $0000_H$ and subsequently continues operation.

#### Conditions

If the TAUBnCMORm.TAUBnMD0 bit is set to 0, the interrupt at start or restart is not generated.

**NOTE**

When TAUBnCMORm.TAUBnCOS[1:0] = $10_B$ or $11_B$, the value of TAUBnCNTm is not written to TAUBnCDRm when the first valid TAUBTTINm input edge occurs after an overflow. However, an interrupt is generated.

#### 24.12.6.2 Equations

TAUBTTINm input pulse interval = count clock cycle ×

[(TAUBnCSRm.TAUBnOVF × ($FFFF_H$ + 1)) + TAUBnCDRm capture value + 1]

### 24.12.6.3 Block Diagram and General Timing Diagram

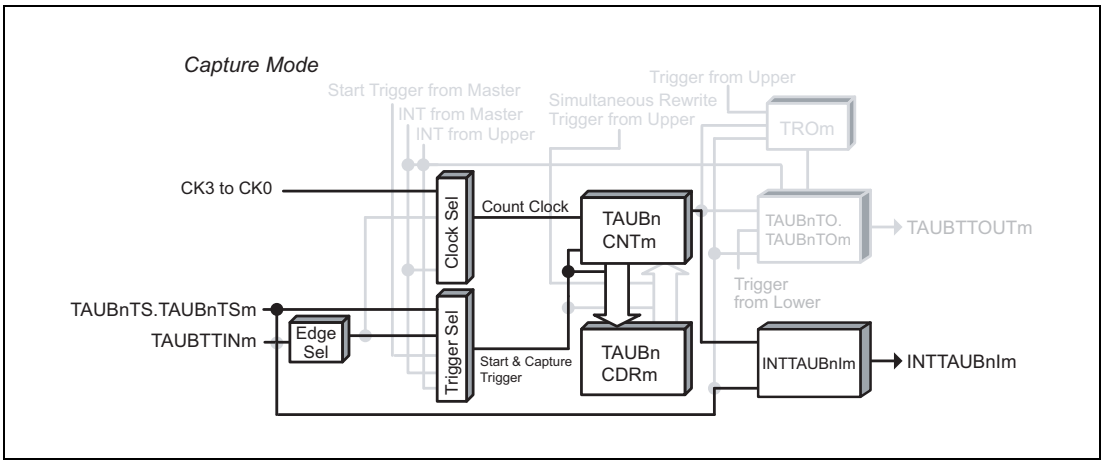



**Figure 24.47 Block Diagram for TAUBTTINm Input Pulse Interval Measurement Function**

The following settings apply to the general timing diagram.

- INTTAUBnIm is not generated at operation start (TAUBnCMORm.TAUBnMD0 = 0)
- Falling edge detection (TAUBnCMURm.TAUBnTIS[1:0] = $00_B$)
- When a valid TAUBTTINm input is detected after an overflow TAUBnCDRm is changed and TAUBnCSRm.TAUBnOVF is set to 1 (TAUBnCMORm.TAUBnCOS[1:0] = $00_B$)

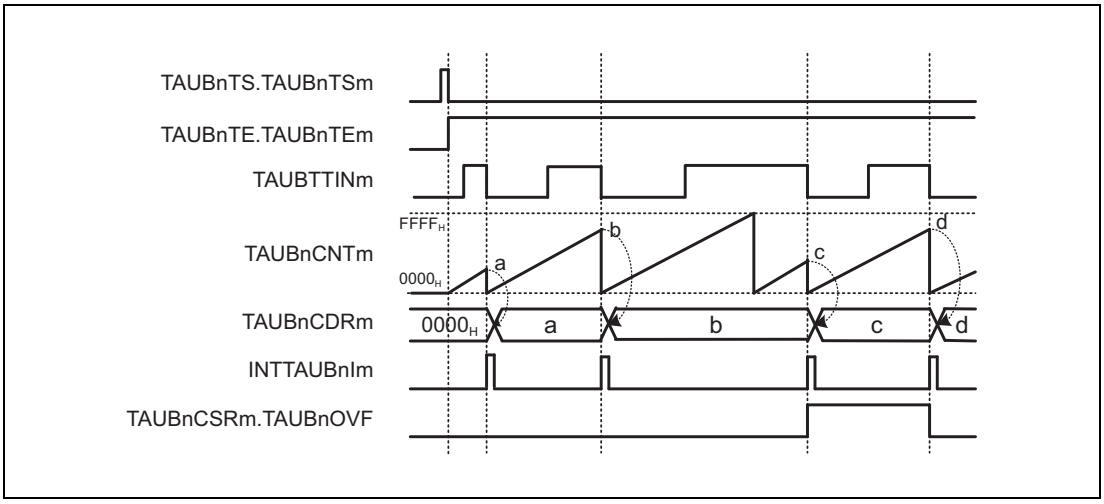



**Figure 24.48 General Timing Diagram for TAUBTTINm Input Pulse Interval Measurement Function**

### 24.12.6.4 Register Settings

#### (1) TAUBnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.61 Contents of the TAUBnCMORm Register for TAUBTTINm Input Pulse Interval Measurement Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $001_B$. |
| 7, 6 | TAUBnCOS[1:0] | See **Table 24.60, Effects of an Overflow** |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0010_B$. |
| 0 | TAUBnMD0 | 0: INTTAUBnIm not generated at operation start<br>1: Generates INTTAUBnIm at operation start |

#### (2) TAUBnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.62 Contents of the TAUBnCMURm Register for TAUBTTINm Input Pulse Interval Measurement Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Falling edge detection<br>01: Rising edge detection<br>10: Rising and falling edge detection<br>11: Setting prohibited |

### (3) Channel output mode

Set TAUBnTOE.TAUBnTOEm to 0 because the channel output mode is not used by this function.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUBnRDE, TAUBnRDS, TAUBnRDM, TAUBnRDC) cannot be used with the TAUBTTINm Input Pulse Interval Measurement Function. Therefore, these registers must be set to 0.

**Table 24.63 Simultaneous Rewrite Settings for TAUBTTINm Input Pulse Interval Measurement Function**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 0: Disables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: When simultaneous rewrite is disabled (TAUBnRDE.TAUBnRDEm = 0), set these bits to 0 |
| TAUBnRDM.TAUBnRDMm | |
| TAUBnRDC.TAUBnRDCm | |

## 24.12.6.5 Operating Procedure for TAUBTTINm Input Pulse Interval Measurement Function

**Table 24.64 Operating Procedure for TAUBTTINm Input Pulse Interval Measurement Function**

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Set the TAUBnCMORm and TAUBnCMURm registers as described in **Table 24.61, Contents of the TAUBnCMORm Register for TAUBTTINm Input Pulse Interval Measurement Function** and **Table 24.62, Contents of the TAUBnCMURm Register for TAUBTTINm Input Pulse Interval Measurement Function**<br><br>The TAUBnCDRm register functions as a capture register. | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm to 1.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is set to 1 and the counter starts.<br>TAUBnCNTm is cleared to $0000_H$. INTTAUBnIm is generated when TAUBnCMORm.TAUBnMD0 is set to 1. |
| During operation | Detection of TAUBTTINm edges.<br><br>The values of TAUBnCMURm.TAUBnTIS[1:0] bits can be changed at any time.<br>The TAUBnCDRm and TAUBnCSRm registers can be read at any time.<br>The TAUBnCSCm.TAUBnCLOV bit can be set to 1. (The TAUBnCSRm.TAUBnOVF bit can be cleared to 0.) | TAUBnCNTm starts to count up from $0000_H$.<br>When a TAUBTTINm valid edge is detected:<br>• TAUBnCNTm transfers (captures) its value to TAUBnCDRm, and returns to $0000_H$<br>• INTTAUBnIm is then generated.<br>Afterwards, this procedure is repeated. |
| Stop operation | Set TAUBnTT.TAUBnTTm to 1.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm stops and both it and TAUBnCSRm.TAUBnOVF retain their current values. |

Restart operation

### 24.12.6.6 Specific Timing Diagrams: Overflow Behavior

#### (1) TAUBnCMORm.TAUBnCOS[1:0] = $00_B$

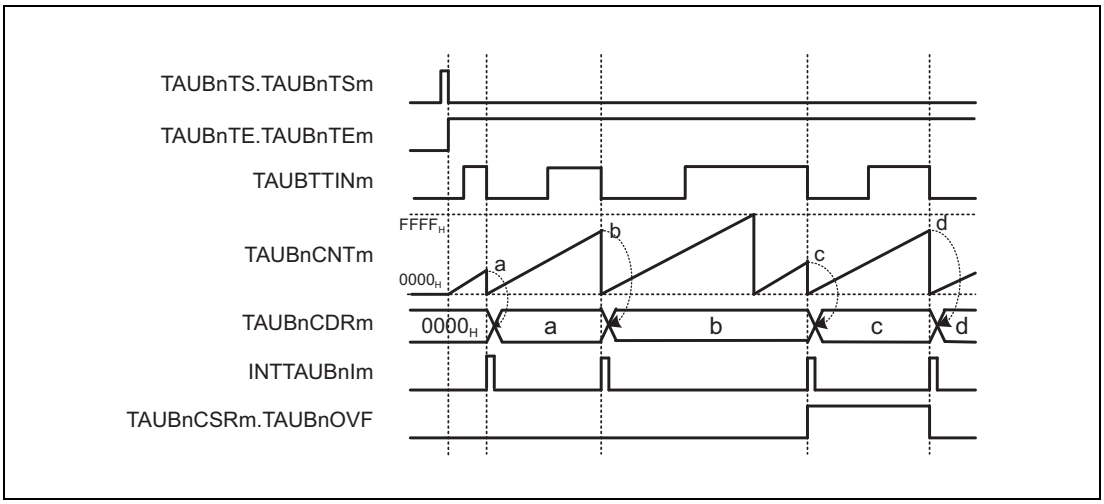



**Figure 24.49 TAUBnCMORm.TAUBnCOS[1:0] = $00_B$, TAUBnCMORm.TAUBnMD0 = 0, TAUBnCMURm.TAUBnTIS[1:0] = $00_B$**

- When an overflow occurs, the value of TAUBnCDRm remains unchanged and TAUBnCSRm.TAUBnOVF remains 0.
- Upon detection of the next valid TAUBTTINm input edge, the value of TAUBnCNTm is loaded to TAUBnCDRm and TAUBnCSRm.TAUBnOVF is set to 1.
- Upon detection of the next valid TAUBTTINm input edge, while any overflow has not occurred, TAUBnCSRm.TAUBnOVF is cleared to 0.

### (2) TAUBnCMORm.TAUBnCOS[1:0] = $01_B$

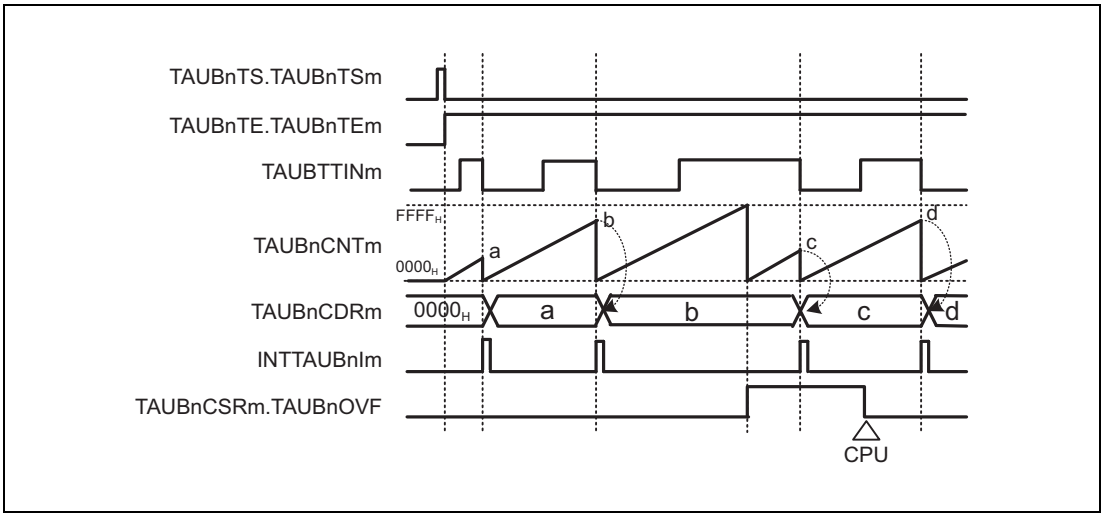



**Figure 24.50 TAUBnCMORm.TAUBnCOS[1:0] = $01_B$, TAUBnCMORm.TAUBnMD0 = 0, TAUBnCMURm.TAUBnTIS[1:0] = $00_B$**

- When an overflow occurs, the value of TAUBnCDRm remains unchanged and TAUBnCSRm.TAUBnOVF is set to 1.
- Upon detection of the next valid TAUBTTINm input edge, the value of TAUBnCNTm is written to TAUBnCDRm.
- TAUBnCSRm.TAUBnOVF is only cleared by a CPU command. (TAUBnCSCm.TAUBnCLOV bit = 1)

### (3) TAUBnCMORm.TAUBnCOS[1:0] = $10_B$

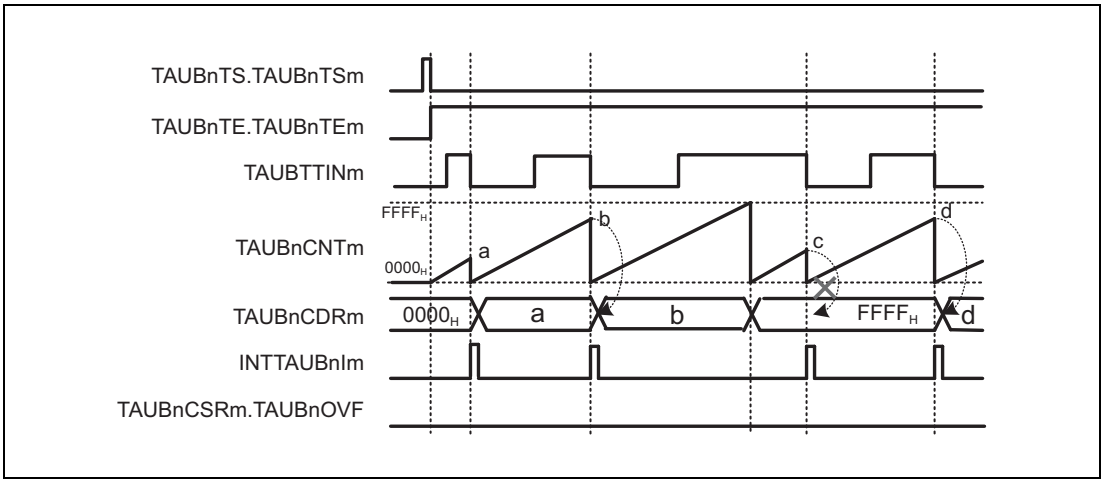



**Figure 24.51 TAUBnCMORm.TAUBnCOS[1:0] = $10_B$, TAUBnCMORm.TAUBnMD0 = 0, TAUBnCMURm.TAUBnTIS[1:0] = $00_B$**

- When an overflow occurs, TAUBnCDRm is set to $FFFF_H$ and TAUBnCSRm.TAUBnOVF remains 0.
- Upon detection of the next valid TAUBTTINm input edge, TAUBnCNTm is reset to 0, but TAUBnCDRm and TAUBnCSRm.TAUBnOVF remain unchanged.
- Thus, the next TAUBTTINm input valid edge after the overflow is ignored.

### (4) TAUBnCMORm.TAUBnCOS[1:0] = $11_B$

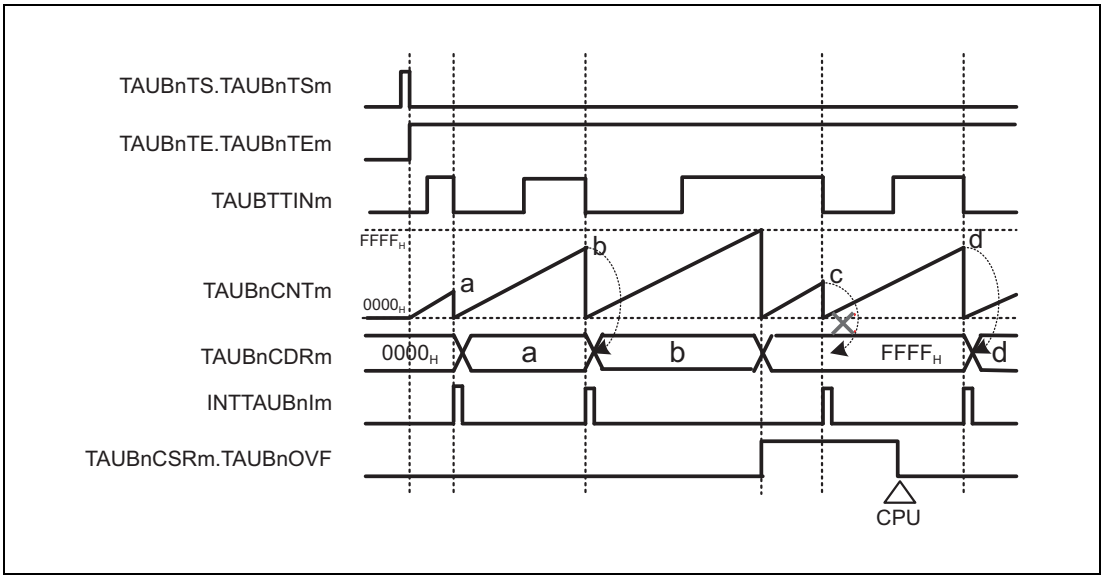



**Figure 24.52 TAUBnCMORm.TAUBnCOS[1:0] = $11_B$, TAUBnCMORm.TAUBnMD0 = 0, TAUBnCMURm.TAUBnTIS[1:0] = $00_B$**

- When an overflow occurs, TAUBnCDRm is set to $FFFF_H$, and TAUBnCSRm.TAUBnOVF is set to 1.
- Upon detection of the next valid TAUBTTINm input edge, TAUBnCNTm is reset to 0, but TAUBnCDRm and TAUBnCSRm.TAUBnOVF remain unchanged.
- Thus, the next TAUBTTINm input valid edge after the overflow is ignored.
- TAUBnCSRm.TAUBnOVF is cleared by setting TAUBnCSCm.TAUBnCLOV = 1.

### (5) When rising and falling edge detection are selected (TAUBnCMORm.TAUBnMD0 = 1)

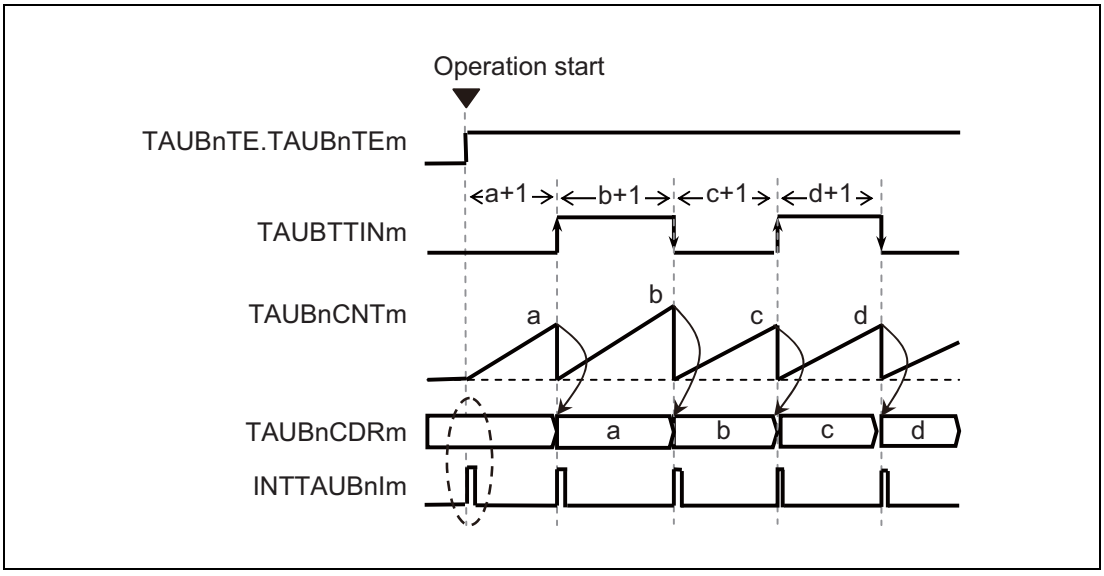



**Figure 24.53 TAUBnCMORm.TAUBnMD0 = 1**

Setting TAUBnCMURm.TAUBnTIS[1:0] to $10_B$ (detection of both edges selected) measures the TAUBTTINm rising and falling edge intervals.

### (6) Operation stop and operation restart (TAUBnCMORm.TAUBnMD0 = 0)

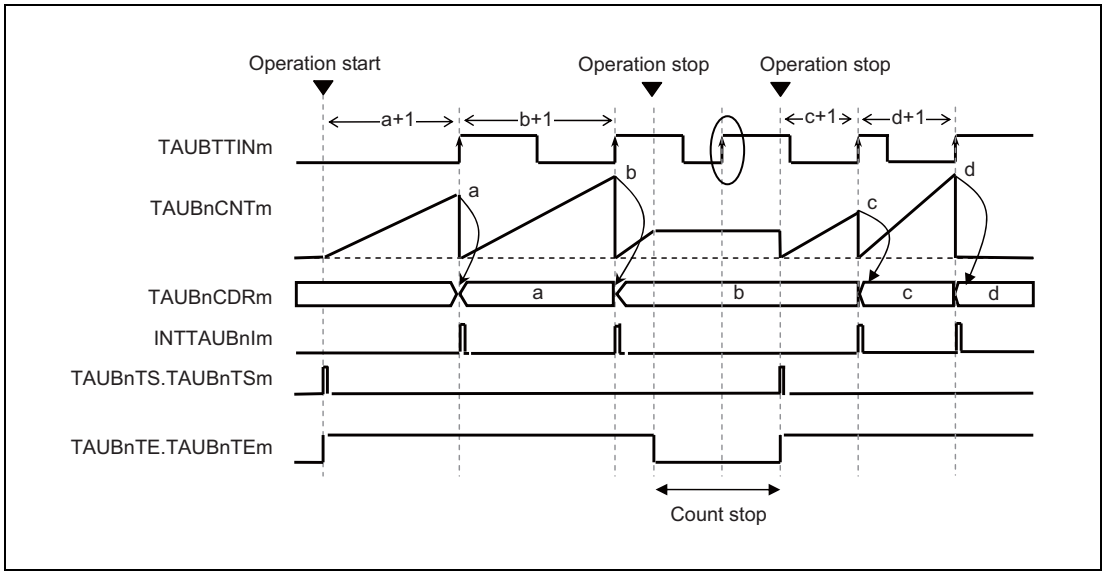



**Figure 24.54 Operation Stop and Operation Restart (TAUBnCMORm.TAUBnMD0 = 0)**

Setting TAUBnTT.TAUBnTTm to 1 clears TAUBnTE.TAUBnTEm to 0, which stops the count operation. At this time, TAUBnCNTm retains the status and stops.

When TAUBnTE.TAUBnTEm retains 0 (operation stopped), TAUBTTINm input is ignored (edge detection is ignored and capture operation is not performed).

Setting TAUBnTS.TAUBnTSm to 1 clears the counter to $0000_H$ and restarts count-up operation.

## 24.12.7 TAUBTTINm Input Signal Width Measurement Function

### 24.12.7.1 Overview

#### Summary

This function measures the width of a TAUBTTINm signal by starting counting on one edge of the TAUBTTINm signal and capturing the counter value on the opposite edge.

#### Prerequisites

TAUBTTOUTm is not used for this function

#### Description

The counter is enabled by setting the channel trigger bit (TAUBnTS.TAUBnTSm) to 1.

This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation. When a valid TAUBTTINm start edge is detected, the counter TAUBnCNTm starts counting up from $0000_H$. When a valid TAUBTTINm stop edge is detected, the value of TAUBnCNTm is captured, transferred to TAUBnCDRm, and an interrupt INTTAUBnIm is generated. The counter retains its value (TAUBnCDRm + 1) and awaits the next valid TAUBTTINm input start edge.

If the counter reaches $FFFF_H$ before a valid TAUBTTINm stop edge is detected, it overflows. The counter is reset to $0000_H$ and subsequently continues operation. The values transferred to TAUBnCDRm and TAUBnCSRm.TAUBnOVF respectively depend on the values of bits TAUBnCMORm.TAUBnCOS[1:0].

**Table 24.65 Effects of an Overflow**

| TAUBnCMORm.TAUBnCOS[1:0] | When Overflow Occurs | | When a Valid TAUBTTINm Input Stop Edge is Detected | |
|---|---|---|---|---|
| | TAUBnCDRm | TAUBnCSRm.TAUBnOVF | TAUBnCDRm, TAUBnCNTm | TAUBnCSRm.TAUBnOVF |
| 00 | Unchanged | 0 | TAUBnCNTm written to TAUBnCDRm | 1 |
| 01 | | 1 | | |
| 10 | Set to $FFFF_H$ | 0 | TAUBnCNTm stops counting TAUBnCDRm unchanged | Unchanged |
| 11 | | 1 | | |

If TAUBnCMORm.TAUBnCOS[0] = 1, the overflow bit TAUBnCSRm.TAUBnOVF can only be cleared by setting TAUBnCSCm.TAUBnCLOV = 1.

The combination of the value of TAUBnCDRm and TAUBnCSRm.TAUBnOVF can be used to deduce the width of the TAUBTTINm signal. However, if an overflow occurs multiple times before a valid TAUBTTINm input is detected, the overflow bit TAUBnCSRm.TAUBnOVF cannot indicate this.

This function cannot be forcibly restarted.

**NOTE**

When TAUBnCMORm.TAUBnCOS[1] = 1, the value of TAUBnCNTm is not written to TAUBnCDRm when the first valid TAUBTTINm input edge occurs after an overflow. However, an interrupt is generated.

### 24.12.7.2 Equations

TAUBTTINm input signal width = count clock cycle ×

[(TAUBnCSRm.OVF × ($FFFF_H$ + 1)) + TAUBnCDRm capture value + 1]

### 24.12.7.3 Block Diagram and General Timing Diagram

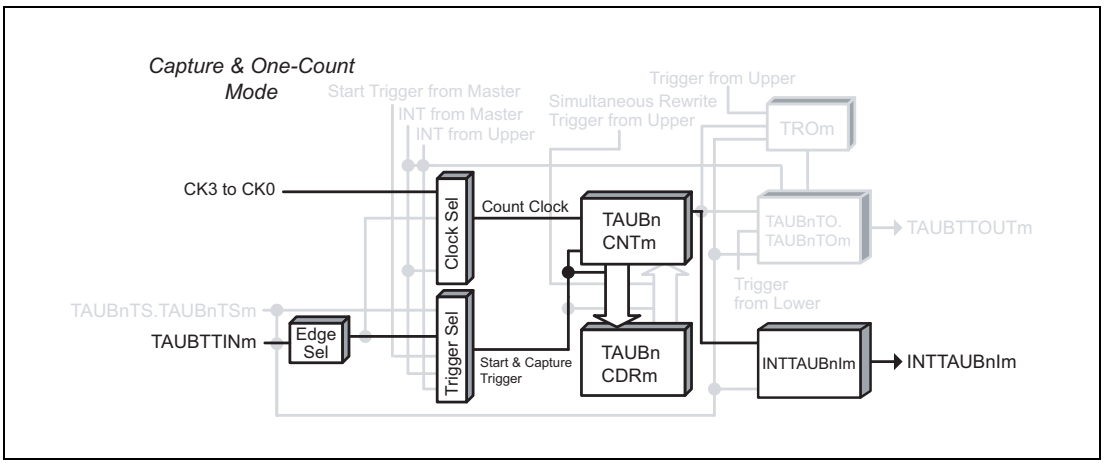



**Figure 24.55 Block Diagram for TAUBTTINm Input Signal Width Measurement Function**

The following settings apply to the general timing diagram.

- Rising and falling edge detection = high width measurement (TAUBnCMURm.TAUBnTIS[1:0] = $11_B$)
- When a valid TAUBTTINm input is detected after an overflow, TAUBnCDRm is changed and TAUBnCSRm.TAUBnOVF is set to 1 (TAUBnCMORm.TAUBnCOS[1:0] = $00_B$)

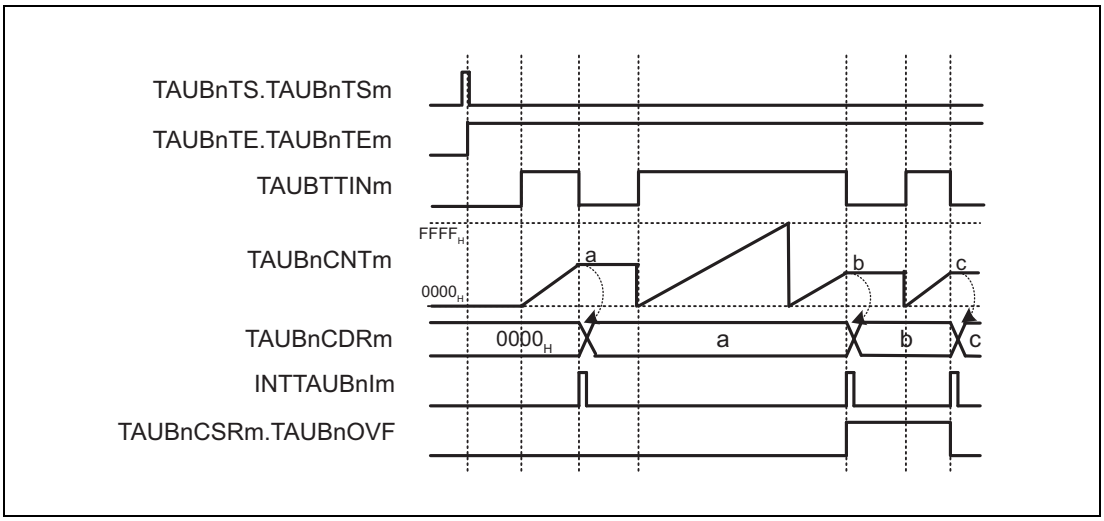



**Figure 24.56 General Timing Diagram for TAUBTTINm Input Signal Width Measurement Function**

### 24.12.7.4 Register Settings

#### (1) TAUBnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.66 Contents of the TAUBnCMORm Register for TAUBTTINm Input Signal Width Measurement Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $010_B$. |
| 7, 6 | TAUBnCOS[1:0] | See **Table 24.65, Effects of an Overflow** |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0110_B$. |
| 0 | TAUBnMD0 | Write $0_B$. |

#### (2) TAUBnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.67 Contents of the TAUBnCMURm Register for TAUBTTINm Input Signal Width Measurement Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 10: Rising and falling edge detection (low width measurement)<br>11: Rising and falling edge detection (high width measurement) |

#### (3) Channel output mode

Set TAUBnTOE.TAUBnTOEm to 0 because the channel output mode is not used by this function.

#### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUBnRDE, TAUBnRDS, TAUBnRDM, and TAUBnRDC) cannot be used with the TAUBTTINm Input Signal Width Measurement Function. Therefore, these registers must be set to 0.

**Table 24.68 Simultaneous Rewrite Settings for TAUBTTINm Input Signal Width Measurement Function**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 0: Disables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: When simultaneous rewrite is disabled (TAUBnRDE.TAUBnRDEm = 0), set these bits to 0 |
| TAUBnRDM.TAUBnRDMm | |
| TAUBnRDC.TAUBnRDCm | |

### 24.12.7.5 Operating Procedure for TAUBTTINm Input Signal Width Measurement Function

**Table 24.69 Operating Procedure for TAUBTTINm Input Signal Width Measurement Function**

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Set the TAUBnCMORm and TAUBnCMURm registers as described in **Table 24.66, Contents of the TAUBnCMORm Register for TAUBTTINm Input Signal Width Measurement Function** and **Table 24.67, Contents of the TAUBnCMURm Register for TAUBTTINm Input Signal Width Measurement Function**<br><br>The TAUBnCDRm register functions as a capture register. | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm to 1.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is set to 1 and TAUBnCNTm waits for detection of the TAUBTTINm start edge.<br>When a TAUBTTINm start edge is detected, TAUBnCNTm start edge to count up. |
| During operation | The TAUBnCDRm, TAUBnCNTm, and TAUBnCSRm registers can be read at any time.<br>TAUBnCSCm.TAUBnCLOV bit can be set to 1. | TAUBnCNTm starts to count up from $0000_H$.<br>When a TAUBTTINm valid edge is detected:<br>• TAUBnCNTm transfers (captures) its value to TAUBnCDRm, and retains its value<br>• INTTAUBnIm is then generated.<br>• The count stops at the value transferred to TAUBnCDRm + 1 and TAUBnCNTm waits for detection of the TAUBTTINm start edge.<br>Afterwards, this procedure is repeated. |
| Stop operation | Set TAUBnTT.TAUBnTTm to 1.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TEm is cleared to 0 and the counter stops.<br>TAUBnCNTm stops and both it and TAUBnCSRm.TAUBnOVF retain their current values. |

Restart operation

### 24.12.7.6 Specific Timing Diagrams: Overflow Behavior

#### (1) TAUBnCMORm.TAUBnCOS[1:0] = $00_B$

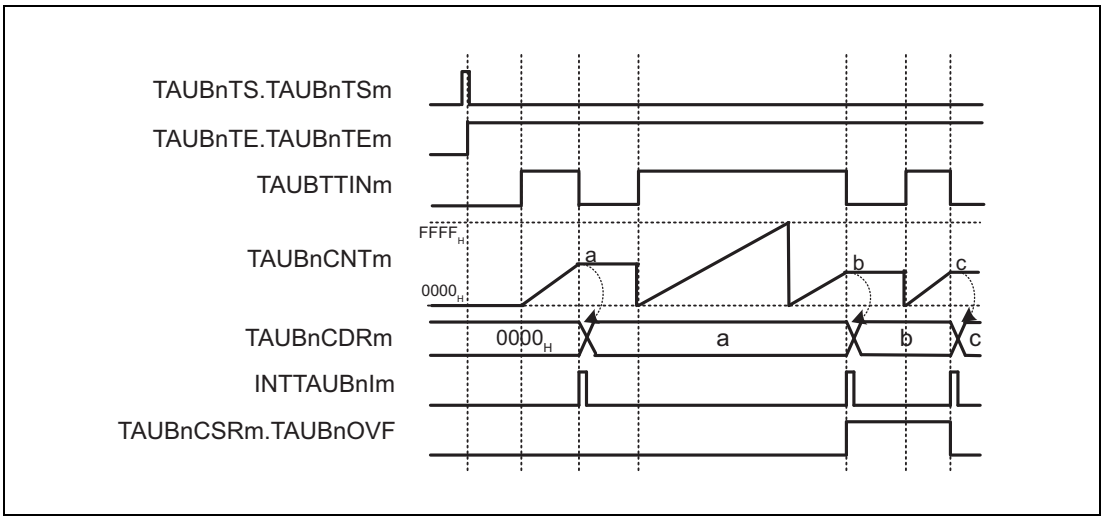



**Figure 24.57 TAUBnCMORm.TAUBnCOS[1:0] = $00_B$, TAUBnCMORm.TAUBnMD0 = 0, TAUBnCMURm.TAUBnTIS[1:0] = $11_B$**

- When an overflow occurs, the value of TAUBnCDRm remains unchanged and TAUBnCSRm.TAUBnOVF remains 0.
- Upon detection of the next valid TAUBTTINm input edge, the value of TAUBnCNTm is written to TAUBnCDRm and TAUBnCSRm.TAUBnOVF is set to 1.
- Upon detection of the next valid TAUBTTINm input edge, while any overflow has not occurred, TAUBnCSRm.TAUBnOVF is cleared to 0.

#### (2) TAUBnCMORm.TAUBnCOS[1:0] = $01_B$

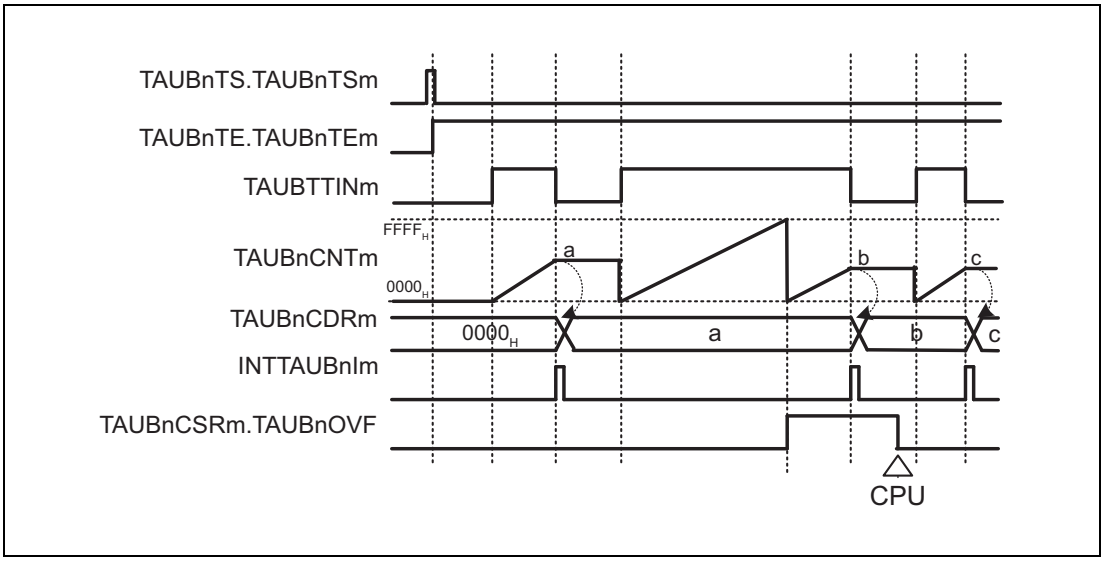



**Figure 24.58 TAUBnCMORm.TAUBnCOS[1:0] = $01_B$, TAUBnCMORm.TAUBnMD0 = 0, TAUBnCMURm.TAUBnTIS[1:0] = $11_B$**

- When an overflow occurs, the value of TAUBnCDRm remains unchanged and TAUBnCSRm.TAUBnOVF is set to 1.
- Upon detection of the next valid TAUBTTINm input edge, the value of TAUBnCNTm is written to TAUBnCDRm.
- TAUBnCSRm.TAUBnOVF is only cleared by a CPU command (The TAUBnCSCm.TAUBnCLOV bit = 1).

### (3) TAUBnCMORm.TAUBnCOS[1:0] = $10_B$

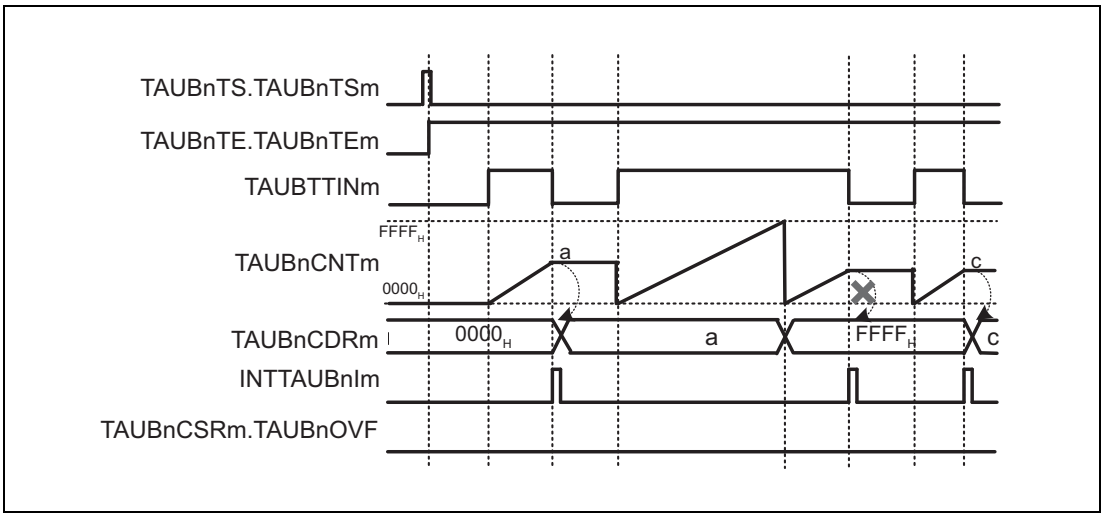



**Figure 24.59 TAUBnCMORm.TAUBnCOS[1:0] = $10_B$, TAUBnCMORm.TAUBnMD0 = 0, TAUBnCMURm.TAUBnTIS[1:0] = $11_B$**

- When an overflow occurs, TAUBnCDRm is set to $FFFF_H$ and TAUBnCSRm.TAUBnOVF remains 0.
- Upon detection of the next valid TAUBTTINm input edge, TAUBnCNTm stops counting, but TAUBnCDRm and TAUBnCSRm.TAUBnOVF remain unchanged.
- Thus, the next TAUBTTINm input valid edge after the overflow is ignored.

### (4) TAUBnCMORm.TAUBnCOS[1:0] = $11_B$

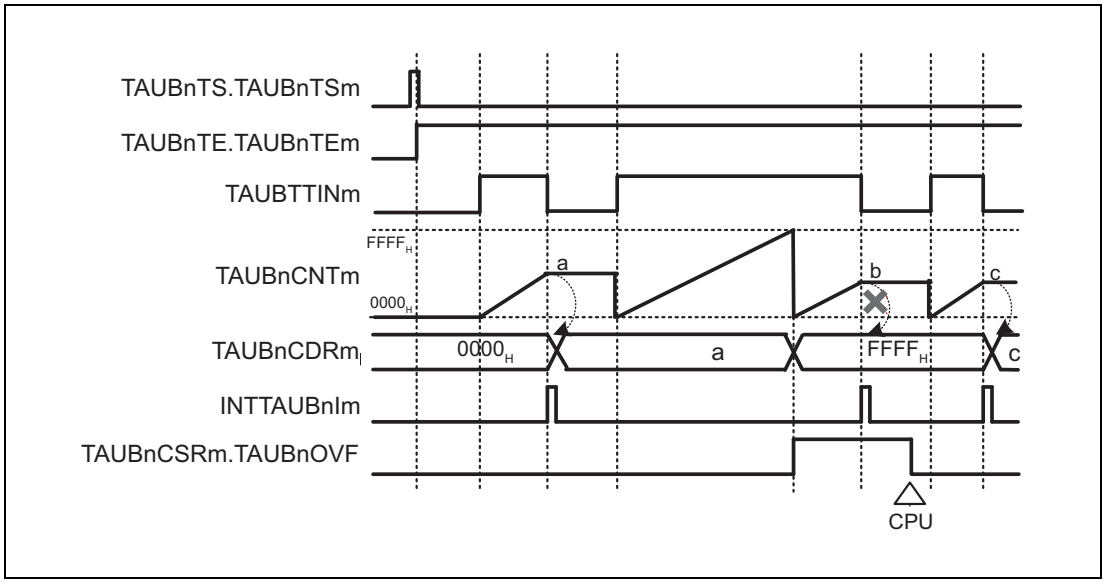



**Figure 24.60 TAUBnCMORm.TAUBnCOS[1:0] = $11_B$, TAUBnCMORm.TAUBnMD0 = 0, TAUBnCMURm.TAUBnTIS[1:0] = $11_B$**

- When an overflow occurs, TAUBnCDRm is set to $FFFF_H$, and TAUBnCSRm.TAUBnOVF is set to 1.
- Upon detection of the next valid TAUBTTINm input edge, TAUBnCNTm stops counting, but TAUBnCDRm and TAUBnCSRm.TAUBnOVF remain unchanged.
- Thus, the next TAUBTTINm input valid edge after the overflow is ignored.
- TAUBnCSRm.TAUBnOVF is cleared by setting TAUBnCSCm.TAUBnCLOV = 1.

### (5) When an overflow occurs (high width measurement)

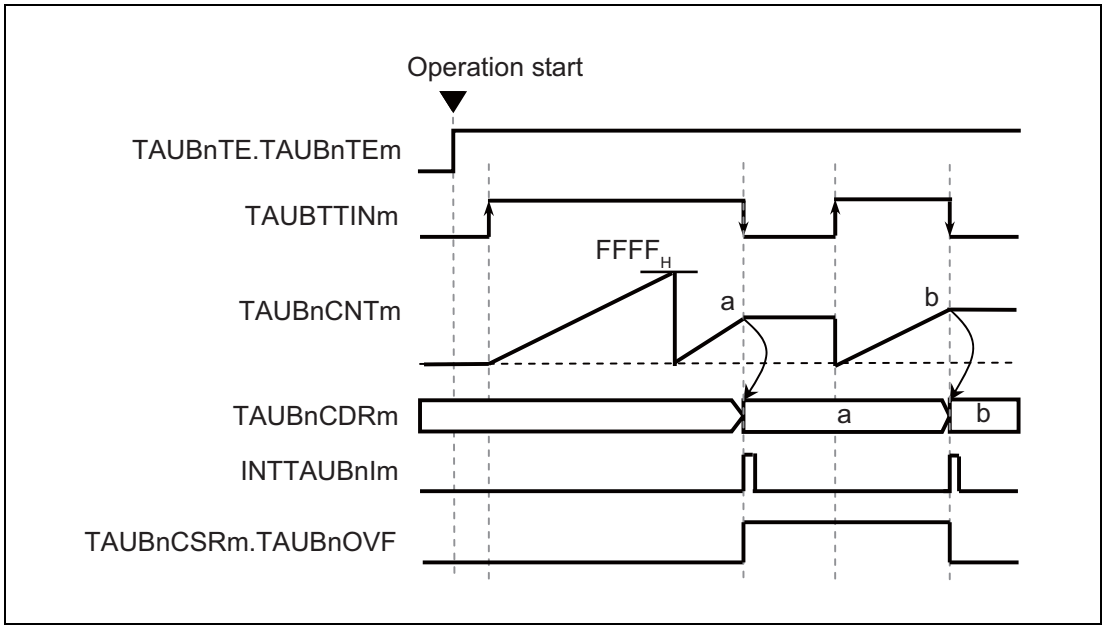



**Figure 24.61 When an Overflow Occurs**

When a capture trigger is input after the counter value has overflowed, the counter value is transferred to TAUBnCDRm and at the same time TAUBnCSRm.TAUBnOVF is set to 1.

TAUBnCSRm.TAUBnOVF is kept at 1 until the next capture trigger occurs.

If the next capture trigger is not accompanied by an overflow, TAUBnCSRm.TAUBnOVF is cleared to 0.

TAUBTTINm input signal width (example when TAUBnCSRm.TAUBnOVF is 1 and TAUBnCDRm is a)
= count clock cycle × (($10000_H$ × TAUBnCSRm.TAUBnOVF) + (TAUBnCDRm capture value + 1))
= count clock cycle × (($10000_H$ × 1) + (a+1))
= count clock cycle × ($10000_H$ + a+1)

## 24.12.8 TAUBTTINm Input Position Detection Function

### 24.12.8.1 Overview

#### Summary

This function measures the interval of input signals by capturing the counter value on a valid edge of the TAUBTTINm signal.

#### Prerequisites

TAUBTTOUTm is not used for this function

#### Description

The counter is enabled by setting the channel trigger bit (TAUBnTS.TAUBnTSm) to 1. This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation. The counter starts to count from $0000_H$. When a valid TAUBTTINm input stop edge is detected, the current TAUBnCNTm value is written to TAUBnCDRm and an interrupt (INTTAUBnIm) is generated. The count operation continues.

When the counter reaches $FFFF_H$, the counter restarts from $0000_H$.

**NOTE**

The TAUBTTINm input signal is sampled at the frequency of the operation clock, specified by the TAUBnCMORm.TAUBnCKS[1:0] bits. As a result, the output cycle of TAUBTTOUTm has an error of ± 1 operation clock cycle.

#### Conditions

If the TAUBnCMORm.MD0 bit is set to 0, the first interrupt after a start or restart is not generated.

### 24.12.8.2 Equations

Function duration at a TAUBTTINm input pulse =

count clock cycle × (TAUBnCDRm capture value + 1)

### 24.12.8.3 Block Diagram and General Timing Diagram

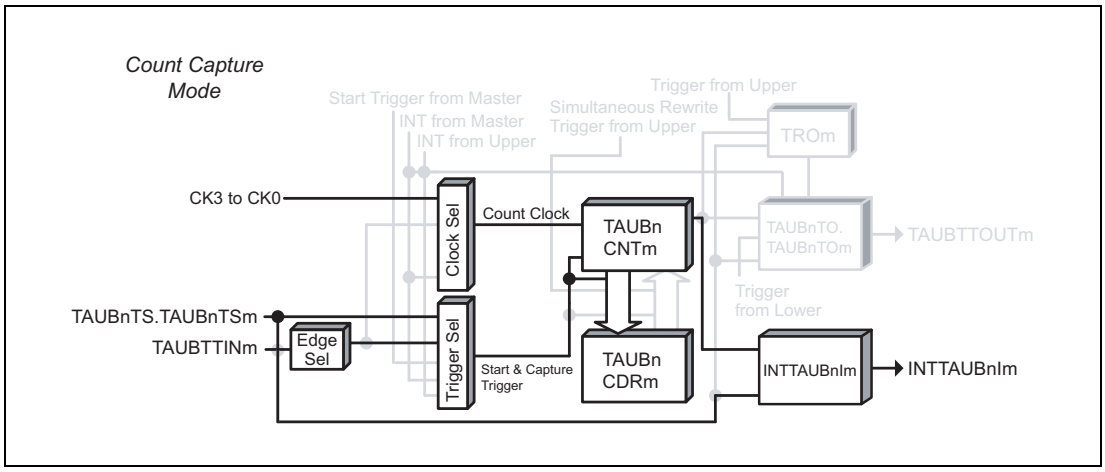



**Figure 24.62 Block Diagram for TAUBTTINm Input Position Detection Function**

The following settings apply to the general timing diagram.

- INTTAUBnIm is not generated at operation start (TAUBnCMORm.TAUBnMD0 = 0)
- Falling edge detection (TAUBnCMURm.TAUBnTIS[1:0] = $00_B$)

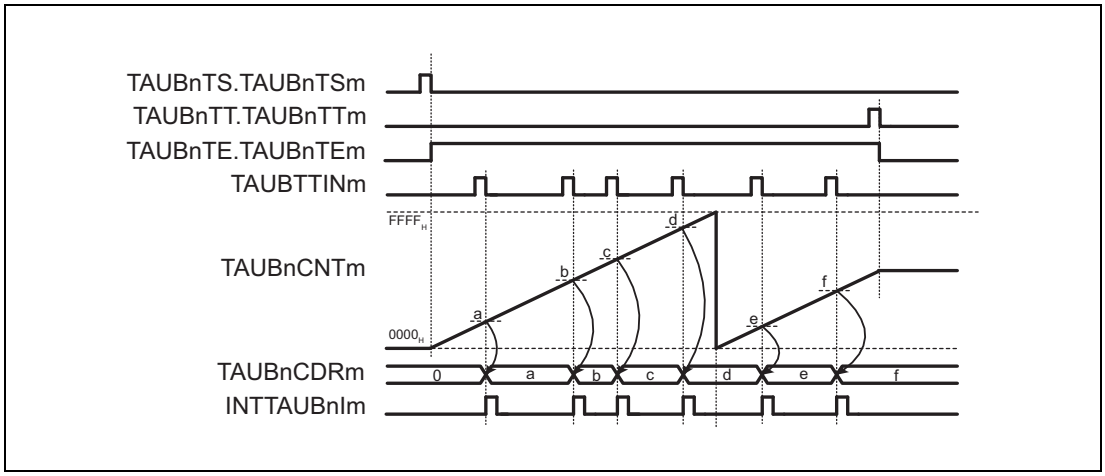



**Figure 24.63 General Timing Diagram for TAUBTTINm Input Position Detection Function**

### 24.12.8.4 Register Settings

#### (1) TAUBnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.70 Contents of the TAUBnCMORm Register for TAUBTTINm Input Position Detection Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $001_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $01_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $1011_B$. |
| 0 | TAUBnMD0 | 0: INTTAUBnIm not generated at operation start<br>1: Generates INTTAUBnIm at operation start |

#### (2) TAUBnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.71 Contents of the TAUBnCMURm Register for TAUBTTINm Input Position Detection Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Falling edge detection<br>01: Rising edge detection<br>10: Rising and falling edge detection<br>11: Setting prohibited |

### (3) Channel output mode

The channel output mode is not used by this function.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUBnRDE, TAUBnRDS, TAUBnRDM, TAUBnRDC) cannot be used with the TAUBTTINm Input Position Detection Function. Therefore, these registers must be set to 0.

**Table 24.72 Simultaneous Rewrite Settings for TAUBTTINm Input Position Detection Function**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 0: Disables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: When simultaneous rewrite is disabled (TAUBnRDE.TAUBnRDEm = 0), set these bits to 0 |
| TAUBnRDM.TAUBnRDMm | |
| TAUBnRDC.TAUBnRDCm | |

## 24.12.8.5 Operating Procedure for TAUBTTINm Input Position Detection Function

**Table 24.73 Operating Procedure for TAUBTTINm Input Position Detection Function**

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Set the TAUBnCMORm and TAUBnCMURm registers as described in **Table 24.70, Contents of the TAUBnCMORm Register for TAUBTTINm Input Position Detection Function** and **Table 24.71, Contents of the TAUBnCMURm Register for TAUBTTINm Input Position Detection Function**<br><br>The TAUBnCDRm register functions as a capture register. | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm to 1.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is set to 1 and the counter starts.<br>INTTAUBnIm is generated when TAUBnCMORm.TAUBnMD0 is set to 1. |
| During operation | The values of TAUBnCMURm.TAUBnTIS[1:0] bits can be changed at any time.<br>The TAUBnCDRm and TAUBnCSRm registers can be read at any time. | TAUBnCNTm starts to count up from $0000_H$.<br>When a TAUBTTINm valid edge is detected:<br>• TAUBnCNTm transfers (captures) its value to TAUBnCDRm<br>• INTTAUBnIm is output.<br>• The counter value is not cleared to $0000_H$ and TAUBnCNTm continues count operation.<br>Afterwards, this procedure is repeated.<br>If TAUBnCNTm reaches $FFFF_H$, the counter restarts from $0000_H$. |
| Stop operation | Set TAUBnTT.TAUBnTTm to 1.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm stops and retains its current value. |

Restart operation

### 24.12.8.6 Specific Timing Diagrams

#### (1) Operation stop and restart

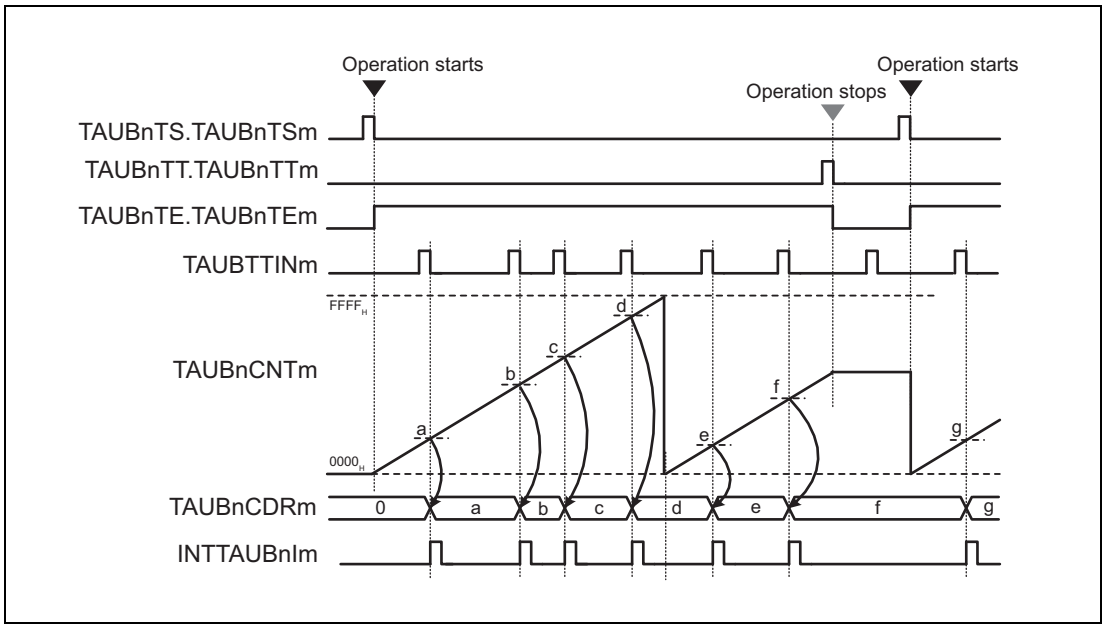



**Figure 24.64 Operation Stop and Restart (TAUBnCMORm.TAUBnMD0 = 0, TAUBnCMURm.TAUBnTIS[1:0] = $00_B$)**

- The counter can be stopped by setting TAUBnTT.TAUBnTTm to 1, which in turn sets TAUBnTE.TEm to 0.
- TAUBnCNTm stops and the current value is retained.
- If the counter is stopped, valid TAUBTTINm input edges are ignored.
- The counter can be restarted by setting TAUBnTS.TAUBnTSm to 1. TAUBnCNTm restarts to count from $0000_H$.

## 24.12.9 TAUBTTINm Input Period Count Detection Function

### 24.12.9.1 Overview

#### Summary

This function measures the cumulative width of a TAUBTTINm input signal.

#### Prerequisites

TAUBTTOUTm is not used for this function

#### Description

The counter is enabled by setting the channel trigger bit (TAUBnTS.TAUBnTSm) to 1.

This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation. The counter awaits a valid TAUBTTINm input edge.

When a valid TAUBTTINm input start edge is detected, the counter starts to count from $0000_H$.

When a valid TAUBTTINm input stop edge is detected, the current TAUBnCNTm value is written to TAUBnCDRm and an interrupt (INTTAUBnIm) is generated. The counter stops and retains its value (TAUBnCDRm + 1) until the next valid TAUBTTINm input start edge is detected.

When a next valid TAUBTTINm input start edge is detected, the counter restarts from the value retained while stopping.

If the counter reaches $FFFF_H$, the counter restarts from $0000_H$.

**NOTES**

1. The TAUBTTINm input signal is sampled at the frequency of the operation clock, specified by the TAUBnCMORm.TAUBnCKS[1:0] bits.
2. As this function is to measure the TAUBTTINm input signal width, setting TAUBnTS.TAUBnTSm to 1 is disabled while TAUBnTE.TAUBnTEm = 1.

#### Conditions

The valid start and stop edges are specified by the TAUBnCMURm.TIS[1:0] bits.

- If TAUBnCMURm.TAUBnTIS[1:0] = $10_B$, the TAUBTTINm input low width is measured. The start trigger is a falling edge and the stop trigger is a rising edge.
- If TAUBnCMURm.TAUBnTIS[1:0] = $11_B$, the TAUBTTINm input high width is measured. The start trigger is a rising edge and the stop trigger is a falling edge.

### 24.12.9.2 Equations

Cumulative TAUBTTINm input width =

count clock cycle × (TAUBnCDRm capture value + 1)

### 24.12.9.3 Block Diagram and General Timing Diagram

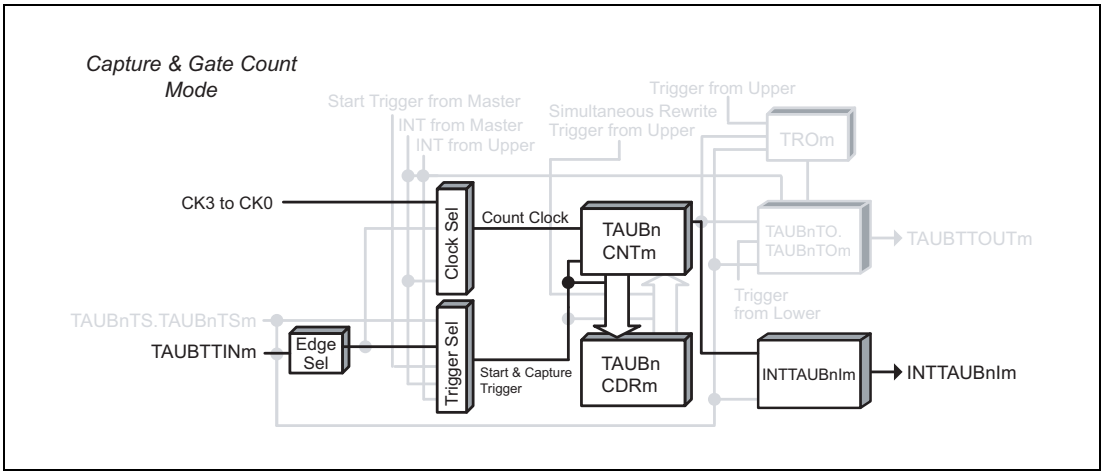



**Figure 24.65 Block Diagram for TAUBTTINm Input Period Count Detection Function**

The following settings apply to the general timing diagram.

- Rising and falling edge detection = high width measurement (TAUBnCMURm.TAUBnTIS[1:0] = $11_B$)

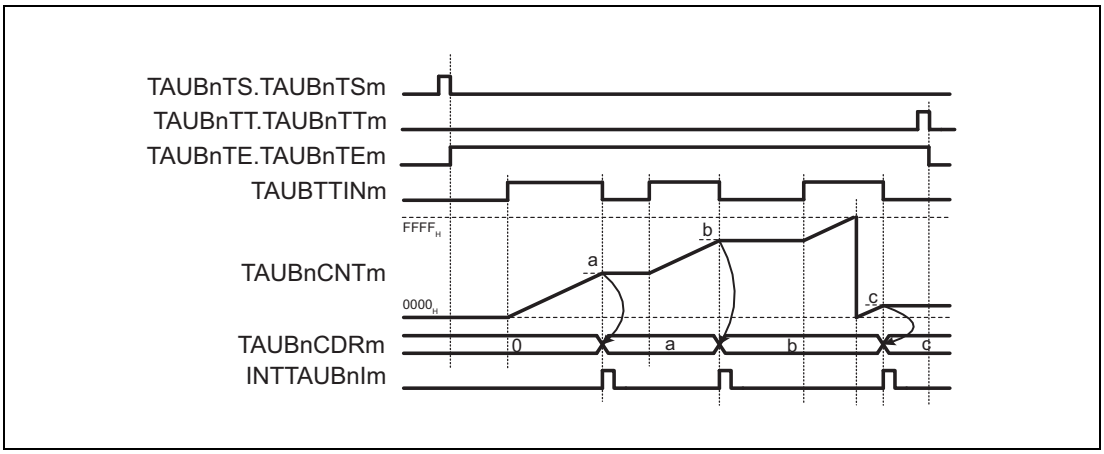



**Figure 24.66 General Timing Diagram for TAUBTTINm Input Period Count Detection Function**

### 24.12.9.4 Register Settings

#### (1) TAUBnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.74 Contents of the TAUBnCMORm Register for TAUBTTINm Input Period Count Detection Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $010_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $01_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $1101_B$. |
| 0 | TAUBnMD0 | Write $0_B$. |

#### (2) TAUBnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.75 Contents of the TAUBnCMURm Register for the TAUBTTINm Input Period Count Detection Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 10: Rising and falling edge detection (Low width measurement)<br>11: Rising and falling edge detection (High width measurement) |

#### (3) Channel output mode

Set TAUBnTOE.TAUBnTOEm to 0 because the channel output mode is not used by this function.

#### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUBnRDE, TAUBnRDS, TAUBnRDM, TAUBnRDC) cannot be used with the TAUBTTINm Input Period Count Detection Function. Therefore, these registers must be set to 0.

**Table 24.76 Simultaneous Rewrite Settings for TAUBTTINm Input Period Count Detection Function**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 0: Disables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: When simultaneous rewrite is disabled (TAUBnRDE.TAUBnRDEm = 0), set these bits to 0 |
| TAUBnRDM.TAUBnRDMm | |
| TAUBnRDC.TAUBnRDCm | |

### 24.12.9.5 Operating Procedure for TAUBTTINm Input Period Count Detection Function

**Table 24.77 Operating Procedure for TAUBTTINm Input Period Count Detection Function**

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Set the TAUBnCMORm and TAUBnCMURm registers as described in **Table 24.74, Contents of the TAUBnCMORm Register for TAUBTTINm Input Period Count Detection Function** and **Table 24.75, Contents of the TAUBnCMURm Register for the TAUBTTINm Input Period Count Detection Function**<br><br>The TAUBnCDRm register functions as a capture register. | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm to 1.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is set to 1 and TAUBnCNTm waits for detection of the TAUBTTINm start edge. |
| During operation | Detection of TAUBTTINm edges.<br><br>The TAUBnCDRm, TAUBnCNTm, and TAUBnCSRm registers can be read at any time. | When a TAUBTTINm start edge (rising edge for high width measurement, falling edge for low width measurement) is detected, TAUBnCNTm starts to count up from the stop value.<br>When TAUBnCNTm detects a stop edge (falling edge for high width measurement, rising edge for low width measurement), it transfers the value to TAUBnCDRm and INTTAUBnIm is generated.<br>Counting stops at the “value transferred to TAUBnCDRm + 1” value and TAUBnCNTm waits for detection of the TAUBTTINm start edge.<br>When TAUBnCNTm reaches $FFFF_H$, the counter restarts from $0000_H$.<br>Afterwards, this procedure is repeated. |
| Stop operation | Set TAUBnTT.TAUBnTTm to 1.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm stops and retains its current value. |

Restart operation

### 24.12.9.6 Specific Timing Diagrams

#### (1) Operation stop and restart

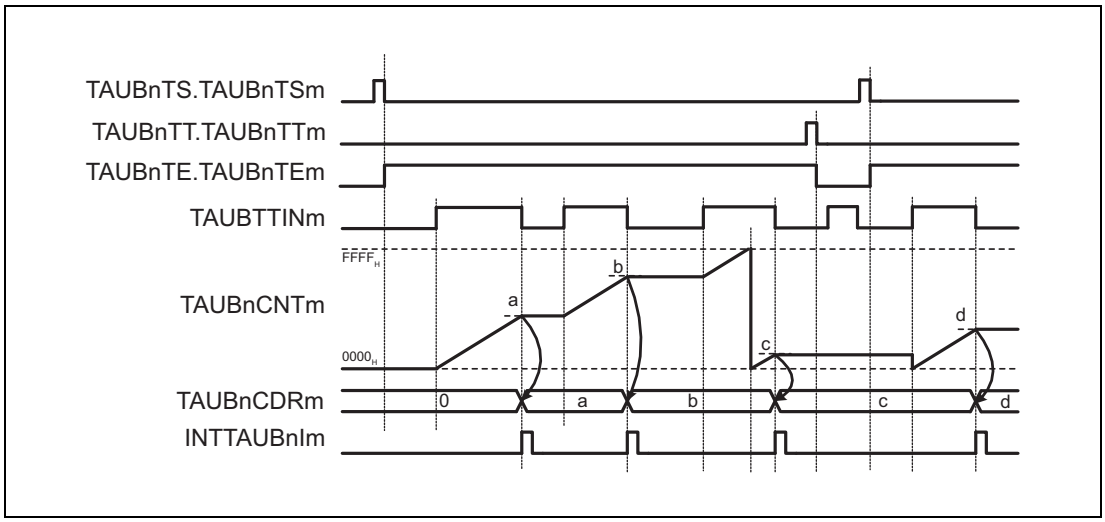



**Figure 24.67 Operation Stop and Restart (TAUBnCMURm.TAUBnTIS[1:0] = $11_B$)**

- The counter can be stopped by setting TAUBnTT.TAUBnTTm to 1, which in turn sets TAUBnTE.TAUBnTEm to 0.
- TAUBnCNTm stops and the current value is retained.
- If the counter is stopped, valid TAUBTTINm input edges are ignored.
- The counter can be restarted by setting TAUBnTS.TAUBnTSm to 1. TAUBnCNTm restarts to count from $0000_H$.

## 24.12.10 TAUBTTINm Input Pulse Interval Judgment Function

### 24.12.10.1 Overview

#### Summary

This function outputs the result of a comparison between the count value (TAUBnCNTm) and the value in the channel data register (TAUBnCDRm) when a TAUBTTINm input pulse occurs. An interrupt request signal INTTAUBnIm is generated if the result of the comparison is true.

#### Prerequisites

TAUBTTOUTm is not used for this function

#### Description

The counter is enabled by setting the channel trigger bit (TAUBnTS.TAUBnTSm) to 1.
This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation. The current value of TAUBnCDRm is written to TAUBnCNTm and the counter starts to count down from this value.

When a TAUBTTINm valid edge is detected or TAUBnTS.TAUBnTSm is set to 1, the function compares the current values of TAUBnCNTm and TAUBnCDRm. An interrupt request signal INTTAUBnIm is generated if the result of the comparison is true. TAUBnCNTm reloads the value of TAUBnCDRm and subsequently continues operation, regardless of the result of the comparison.

If the counter reaches $0000_H$ before a TAUBTTINm valid edge is detected, TAUBnCNTm overflows and is set to $FFFF_H$. It then continues to count down.

The value of TAUBnCDRm can be rewritten at any time, and the changed value of TAUBnCDRm is applied the next time the function starts to count down.

#### Conditions

The TAUBnCMORm.TAUBnMD0 bit specifies the type of comparison:

- If TAUBnCMORm.TAUBnMD0 = 0, INTTAUBnIm is generated when TAUBnCNTm ≤ TAUBnCDRm.
- If TAUBnCMORm.TAUBnMD0 = 1, INTTAUBnIm is generated when TAUBnCNTm > TAUBnCDRm.

### 24.12.10.2 Block Diagram and General Timing Diagram

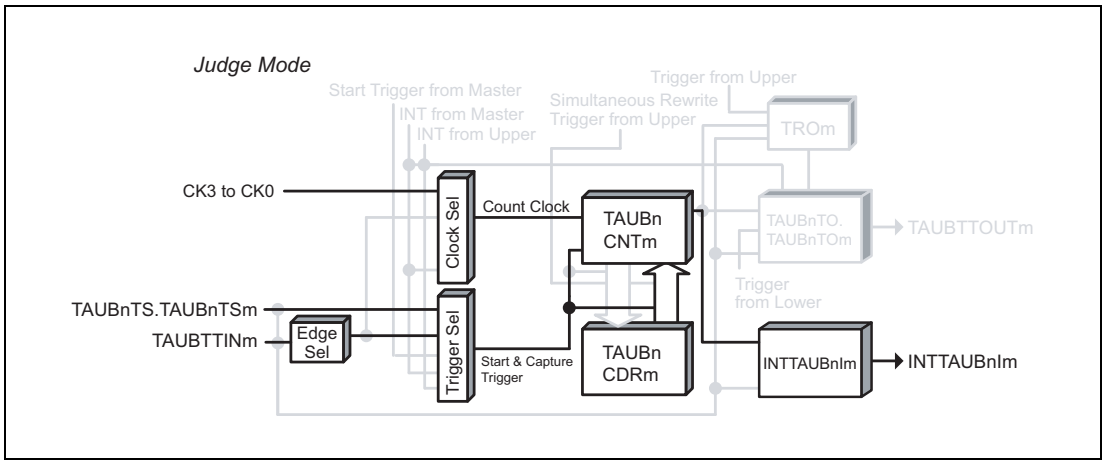



**Figure 24.68 Block Diagram for TAUBTTINm Input Pulse Interval Judgment Function**

The following settings apply to the general timing diagram.

- Falling edge detection (TAUBnCMURm.TAUBnTIS[1:0] = $00_B$)

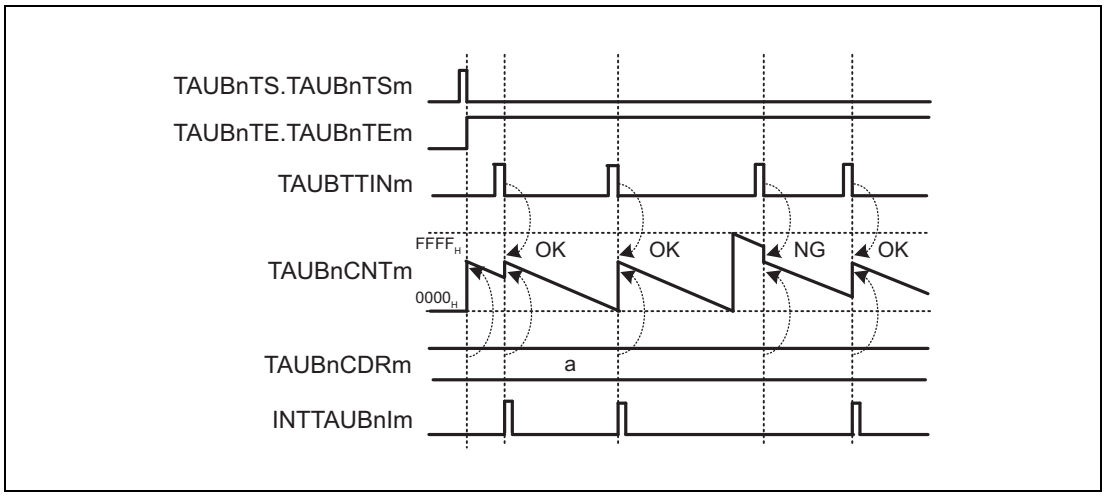



**Figure 24.69 General Timing Diagram for TAUBTTINm Input Pulse Interval Judgment Function**

### 24.12.10.3 Register Settings

#### (1) TAUBnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.78 Contents of the TAUBnCMORm Register for TAUBTTINm Input Pulse Interval Judgment Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $001_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0001_B$. |
| 0 | TAUBnMD0 | 0: INTTAUBnIm is generated when TAUBnCNTm ≤ TAUBnCDRm<br>1: INTTAUBnIm is generated when TAUBnCNTm > TAUBnCDRm |

### (2) TAUBnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.79 Contents of the TAUBnCMURm Register for TAUBTTINm Input Pulse Interval Judgment Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Falling edge detection<br>01: Rising edge detection<br>10: Rising and falling edge detection<br>11: Setting prohibited |

### (3) Channel output mode

Set TAUBnTOE.TAUBnTOEm to 0 because the channel output mode is not used by this function.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUBnRDE, TAUBnRDS, TAUBnRDM, and TAUBnRDC) cannot be used with the TAUBTTINm Input Pulse Interval Judgment Function. Therefore, these registers must be set to 0.

**Table 24.80 Simultaneous Rewrite Settings for TAUBTTINm Input Pulse Interval Judgment Function**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 0: Disables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: When simultaneous rewrite is disabled (TAUBnRDE.TAUBnRDEm = 0), sets these bits to 0 |
| TAUBnRDM.TAUBnRDMm | |
| TAUBnRDC.TAUBnRDCm | |

## 24.12.10.4 Operating Procedure for TAUBTTINm Input Pulse Interval Judgment Function

**Table 24.81 Operating Procedure for TAUBTTINm Input Pulse Interval Judgment Function**

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Set the TAUBnCMORm and TAUBnCMURm registers as described in **Table 24.78, Contents of the TAUBnCMORm Register for TAUBTTINm Input Pulse Interval Judgment Function** and **Table 24.79, Contents of the TAUBnCMURm Register for TAUBTTINm Input Pulse Interval Judgment Function**<br><br>Set the value of the TAUBnCDRm register | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm to 1.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is set to 1 and the counter starts.<br>TAUBnCNTm loads the TAUBnCDRm value. |
| During operation | The following register can be changed at any time:<br>• TAUBnCDRm register | When TAUBnCMORm.TAUBnMD0 = 0<br>If TAUBnCNTm ≤ TAUBnCDRm when a TAUBTTINm input edge is detected, INTTAUBnIm is generated.<br><br>When TAUBnCMORm.TAUBnMD0 = 1<br>If TAUBnCNTm > TAUBnCDRm when a TAUBTTINm input edge is detected, INTTAUBnIm is generated. If a TAUBTTINm input edge is detected, then TAUBnCNTm starts to count down from the value of TAUBnCDRm.<br><br>Afterwards, this procedure is repeated. |
| Stop operation | Set TAUBnTT.TAUBnTTm to 1.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm stops and retains its current value. |

Restart operation

## 24.12.11 TAUBTTINm Input Signal Width Judgment Function

### 24.12.11.1 Overview

#### Summary

This function compares the count value (TAUBnCNTm) for the high or low level width of a TAUBTTINm input signal and the TAUBnCDRm value, and outputs the judgment result from the interrupt request signal INTTAUBnIm.

#### Prerequisites

TAUBTTOUTm is not used for this function

#### Description

The counter is enabled by setting the channel trigger bit (TAUBnTS.TAUBnTSm) to 1. This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation. When a valid TAUBTTINm input start edge is detected, the current value of TAUBnCDRm is written to TAUBnCNTm and the counter starts to count down from this value.

When a TAUBTTINm valid stop edge is detected, the function compares the current values of TAUBnCNTm and TAUBnCDRm. An interrupt request signal INTTAUBnIm is generated if the result of the comparison is true. The counter TAUBnCNTm retains its value until the next TAUBTTINm valid start edge is detected, regardless of the result of the comparison.

If the counter reaches $0000_H$ before a valid TAUBTTINm stop edge is detected, TAUBnCNTm overflows and is set to $FFFF_H$. It then continues to count down.

The value of TAUBnCDRm can be rewritten at any time, and the changed value of TAUBnCDRm is applied the next time the function starts to count down.

#### Conditions

- The TAUBnCMORm.TAUBnMD0 bit specifies the type of comparison:
  - If TAUBnCMORm.TAUBnMD0 = 0, INTTAUBnIm is generated when TAUBnCNTm ≤ TAUBnCDRm.
  - If TAUBnCMORm.TAUBnMD0 = 1, INTTAUBnIm is generated when TAUBnCNTm > TAUBnCDRm.
- The TAUBnCMURm.TAUBnTIS[1:0] bits specify the type of width measurement:
  - For high width measurement, (TAUBnCMURm.TAUBnTIS[1:0] = $11_B$) the start edge is a rising TAUBTTINm edge and the stop edge is a falling TAUBTTINm edge.
  - For low width measurement, (TAUBnCMURm.TAUBnTIS[1:0] = $10_B$) the start edge is a falling TAUBTTINm edge and the stop edge is a rising TAUBTTINm edge.
- Forced restart is not possible for this function.

### 24.12.11.2 Block Diagram and General Timing Diagram

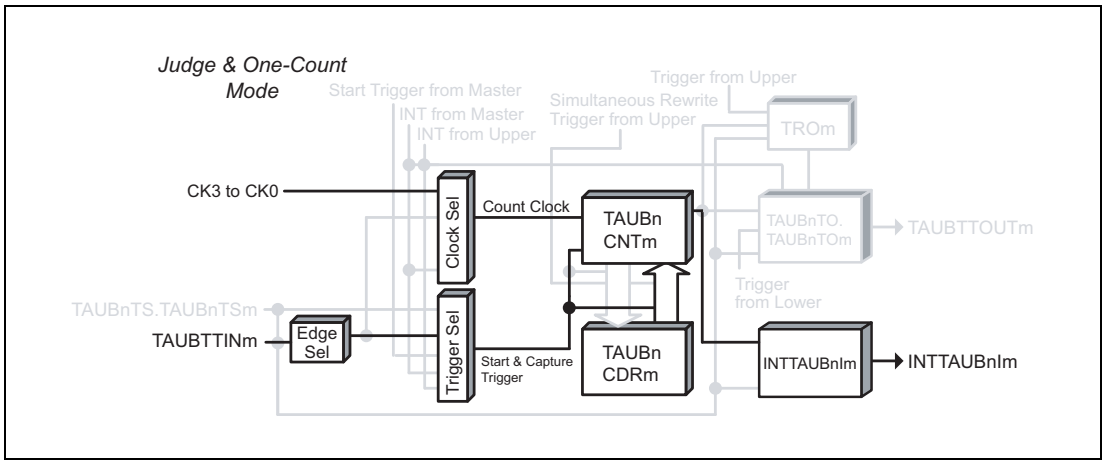



**Figure 24.70 Block Diagram for TAUBTTINm Input Signal Width Judgment Function**

The following settings apply to the general timing diagram.

- INTTAUBnIm is generated when TAUBnCNTm ≤ TAUBnCDRm (TAUBnCMORm.TAUBnMD0 = 0)
- TAUBTTINm valid start edge = rising edge, TAUBTTINm valid stop edge = falling edge (TAUBnCMURm.TAUBnTIS[1:0] = $11_B$)

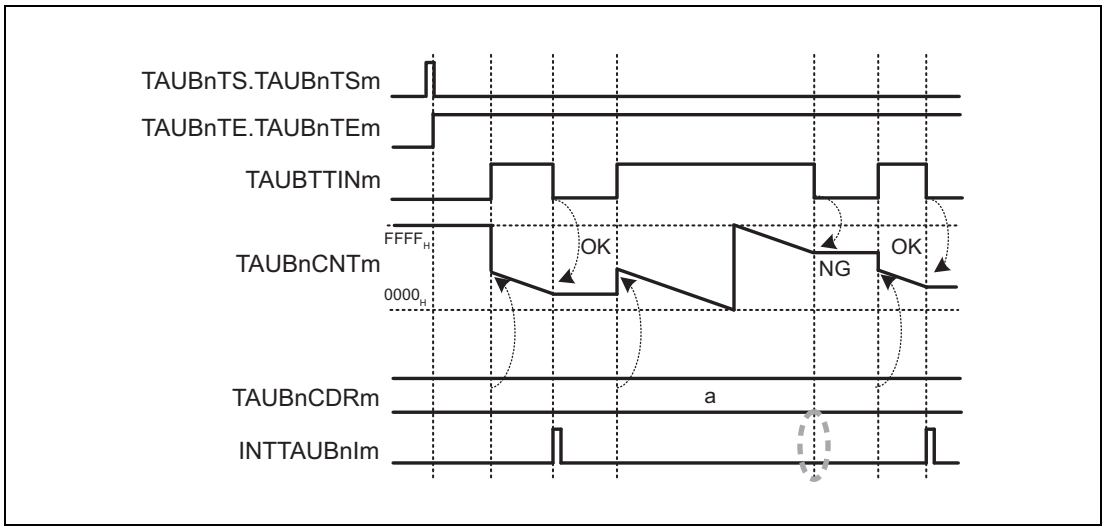



**Figure 24.71 General Timing Diagram for TAUBTTINm Input Signal Width Judgment Function**

### 24.12.11.3 Register Settings

#### (1) TAUBnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.82 Contents of the TAUBnCMORm Register for TAUBTTINm Input Signal Width Judgment Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $010_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0111_B$. |
| 0 | TAUBnMD0 | 0: INTTAUBnIm is generated when TAUBnCNTm ≤ TAUBnCDRm<br>1: INTTAUBnIm is generated when TAUBnCNTm > TAUBnCDRm |

#### (2) TAUBnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.83 Contents of the TAUBnCMURm Register for TAUBTTINm Input Signal Width Judgment Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 10: Rising and falling edge detection (low width measurement)<br>11: Rising and falling edge detection (high width measurement) |

### (3) Channel output mode

Set TAUBnTOE.TAUBnTOEm to 0 because the channel output mode is not used by this function.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUBnRDE, TAUBnRDS, TAUBnRDM, and TAUBnRDC) cannot be used with the TAUBTTINm Input Signal Width Judgment Function. Therefore, these registers must be set to 0.

**Table 24.84 Simultaneous Rewrite Settings for TAUBTTINm Input Signal Width Judgment Function**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 0: Disables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: When simultaneous rewrite is disabled (TAUBnRDE.TAUBnRDEm = 0), sets these bits to 0 |
| TAUBnRDM.TAUBnRDMm | |
| TAUBnRDC.TAUBnRDCm | |

## 24.12.11.4 Operating Procedure for TAUBTTINm Input Signal Width Judgment Function

**Table 24.85 Operating Procedure for TAUBTTINm Input Signal Width Judgment Function**

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Set the TAUBnCMORm and TAUBnCMURm registers as described in **Table 24.82, Contents of the TAUBnCMORm Register for TAUBTTINm Input Signal Width Judgment Function** and **Table 24.83, Contents of the TAUBnCMURm Register for TAUBTTINm Input Signal Width Judgment Function**<br><br>Set the value of the TAUBnCDRm register | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm to 1.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is set to 1 and TAUBnCNTm waits for detection of the TAUBTTINm start edge. |
| During operation | The following register can be changed at any time:<br>• TAUBnCDRm register | If a TAUBTTINm start edge is detected, then TAUBnCNTm starts to count down from the value of TAUBnCDRm.<br><br>When TAUBnCMORm.TAUBnMD0 = 0<br>If TAUBnCNTm ≤ TAUBnCDRm when a TAUBTTINm input stop edge is detected, INTTAUBnIm is generated.<br><br>When TAUBnCMORm.TAUBnMD0 = 1<br>If TAUBnCNTm > TAUBnCDRm when a TAUBTTINm input stop edge is detected, INTTAUBnIm is generated.<br><br>Afterwards, this procedure is repeated. |
| Stop operation | Set TAUBnTT.TAUBnTTm to 1.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm stops and retains its current value. |

Restart operation

## 24.12.12 Overflow Interrupt Output Function (during TAUBTINm Width Measurement)

### 24.12.12.1 Overview

#### Summary

This function measures the width of an individual TAUBTTINm input signal. An interrupt is generated if the TAUBTTINm input width is longer than $FFFF_H + 1$.

#### Prerequisites

- TAUBTTOUTm is not used for this function
- The value of TAUBnCDRm must be set to $FFFF_H$.

#### Description

The counter is enabled by setting the channel trigger bit (TAUBnTS.TAUBnTSm) to 1. This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation.

The counter starts when a valid TAUBTTINm input start edge is detected. $FFFF_H$ is written to TAUBnCNTm and the counter starts to count down.

When a valid stop edge is detected, the counter stops and retains the current value.

When the next TAUBTTINm input start edge is detected, TAUBnCNTm loads $FFFF_H$ and starts to count down.

If the counter reaches $0000_H$ before a stop edge is detected, an interrupt is generated.

#### Conditions

The valid start and stop edges are specified by the TAUBnCMURm.TAUBnTIS[1:0] bits.

- If TAUBnCMURm.TAUBnTIS[1:0] = $10_B$, the TAUBTTINm input low width is measured. The start trigger is a falling edge and the stop trigger is a rising edge.
- If TAUBnCMURm.TAUBnTIS[1:0] = $11_B$, the TAUBTTINm input high width is measured. The start trigger is a rising edge and the stop trigger is a falling edge.

**NOTE**

The counter cannot be restarted during operation.

### 24.12.12.2 Block Diagram and General Timing Diagram

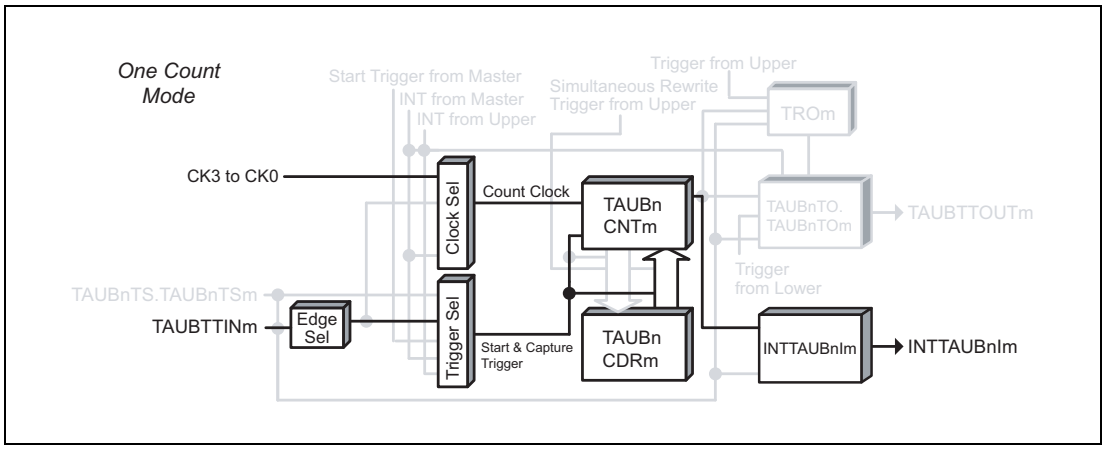



**Figure 24.72 Block Diagram for Overflow Interrupt Output Function (during TAUBTTINm Width Measurement)**

The following settings apply to the general timing diagram.

- Rising and falling edge detection = high width measurement (TAUBnCMURm.TAUBnTIS[1:0] = $11_B$)

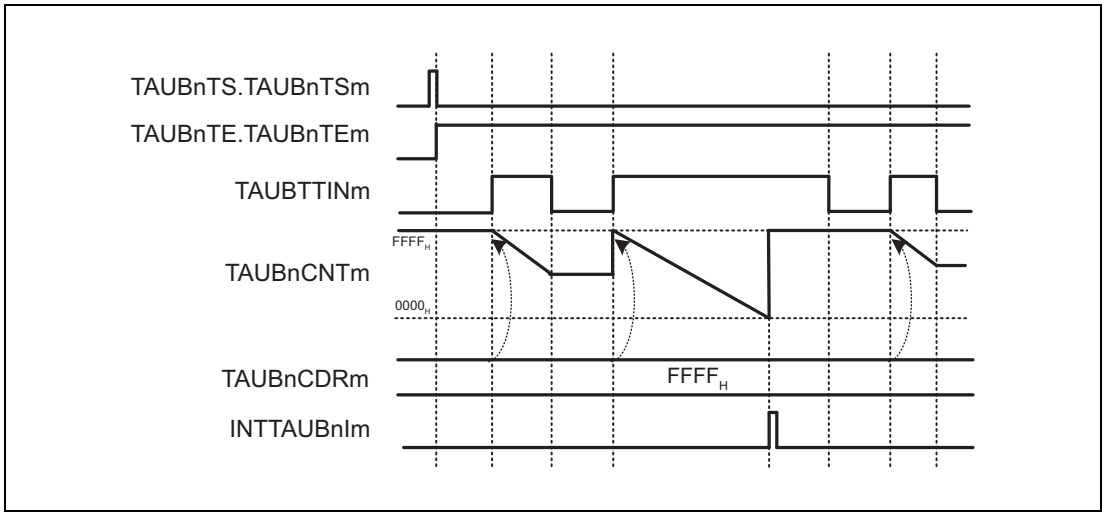



**Figure 24.73 General Timing Diagram for Overflow Interrupt Output Function (during TAUBTTINm Width Measurement)**

### 24.12.12.3 Register Settings

#### (1) TAUBnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.86 Contents of the TAUBnCMORm Register for Overflow Interrupt Output Function (during TAUBTTINm Width Measurement)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $010_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $0_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0100_B$. |
| 0 | TAUBnMD0 | Write $0_B$. |

#### (2) TAUBnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.87 Contents of the TAUBnCMURm Register for Overflow Interrupt Output Function (during TAUBTTINm Width Measurement)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 10: Rising and falling edge detection (Low width measurement)<br>11: Rising and falling edge detection (High width measurement) |

### (3) Channel output mode

Set TAUBnTOE.TAUBnTOEm to 0 because the channel output mode is not used by this function.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUBnRDE, TAUBnRDS, TAUBnRDM, and TAUBnRDC) cannot be used with the Overflow Interrupt Output Function (during TAUBTTINm Width Measurement). Therefore, these registers must be set to 0.

**Table 24.88 Simultaneous Rewrite Settings for Overflow Interrupt Output Function (during TAUBTTINm Width Measurement)**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 0: Disables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: When simultaneous rewrite is disabled (TAUBnRDE.TAUBnRDEm = 0), set these bits to 0 |
| TAUBnRDM.TAUBnRDMm | |
| TAUBnRDC.TAUBnRDCm | |

### 24.12.12.4 Operating Procedure for Overflow Interrupt Output Function (during TAUBTTINm Width Measurement)

Table 24.89 Operating Procedure for Overflow Interrupt Output Function (during TAUBTTINm Width Measurement)

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Set the TAUBnCMORm and TAUBnCMURm registers as described in **Table 24.86, Contents of the TAUBnCMORm Register for Overflow Interrupt Output Function (during TAUBTTINm Width Measurement)** and **Table 24.87, Contents of the TAUBnCMURm Register for Overflow Interrupt Output Function (during TAUBTTINm Width Measurement)**<br><br>Set the value of the TAUBnCDRm register to $FFFF_H$. | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm to 1.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0.<br><br>Detection of TAUBTTINm start edge | TAUBnTE.TAUBnTEm is set to 1 and TAUBnCNTm waits for detection of the start edge.<br><br>When a start edge is detected, TAUBnCNTm loads the TAUBnCDRm value ($FFFF_H$). |
| During operation | The TAUBnCNTm register can be read at any time. | TAUBnCNTm counts down. When the counter reaches $0000_H$:<br>• INTTAUBnIm is generated<br>When a TAUBTTINm input stop edge is detected during count operation:<br>• TAUBnCNTm stops and retains its current value.<br>When a TAUBTTINm input start edge is detected while the counter is stopped:<br>• The TAUBnCDRm value ($FFFF_H$) is loaded to TAUBnCNTm again and the counter starts to count down.<br>Afterwards, this procedure is repeated. |
| Stop operation | Set TAUBnTT.TAUBnTTm to 1.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm stops and retains its current value. |

Restart operation

## 24.12.13 Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection)

### 24.12.13.1 Overview

#### Summary

This function measures the cumulative width of a TAUBTTINm input signal. An interrupt is generated and an overflow interrupt can be output if the cumulative TAUBTTINm input width is longer than $FFFF_H$.

#### Prerequisites

- TAUBTTOUTm is not used for this function
- The value of TAUBnCDRm must be set to $FFFF_H$

#### Description

The counter is enabled by setting the channel trigger bit (TAUBnTS.TAUBnTSm) to 1. This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation.

The counter starts when a valid TAUBTTINm input start edge is detected. $FFFF_H$ is written to TAUBnCNTm and the counter starts to count down.

When a valid stop edge is detected, the counter stops and retains the current value. The counter awaits the next TAUBTTINm input start edge and then continues to count down from the current value.

When the counter reaches $0000_H$ an interrupt is generated. $FFFF_H$ is written to TAUBnCNTm and the counter continues to count down until a TAUBTTINm input stop edge is detected.

#### Conditions

The valid start and stop edges are specified by the TAUBnCMURm.TAUBnTIS[1:0] bits.

- If TAUBnCMURm.TAUBnTIS[1:0] = $10_B$, the TAUBTTINm input low width is counted. The start trigger is a falling edge and the stop trigger is a rising edge.
- If TAUBnCMURm.TAUBnTIS[1:0] = $11_B$, the TAUBTTINm input high width is counted. The start trigger is a rising edge and the stop trigger is a falling edge.

**NOTE**

The counter cannot be restarted during operation.

### 24.12.13.2 Block Diagram and General Timing Diagram

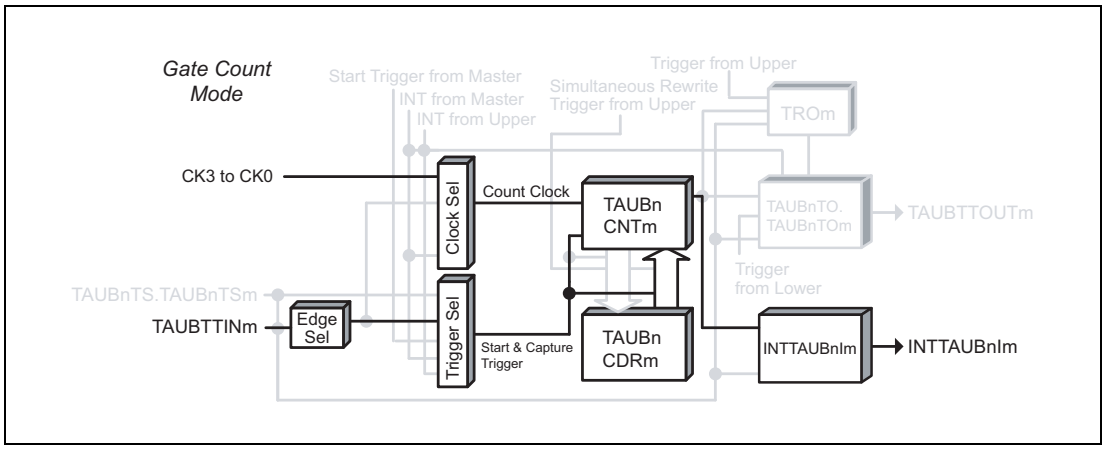



**Figure 24.74 Block Diagram for Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection)**

The following settings apply to the general timing diagram.

- Rising and falling edge detection = high width measurement (TAUBnCMURm.TAUBnTIS[1:0] = $11_B$)

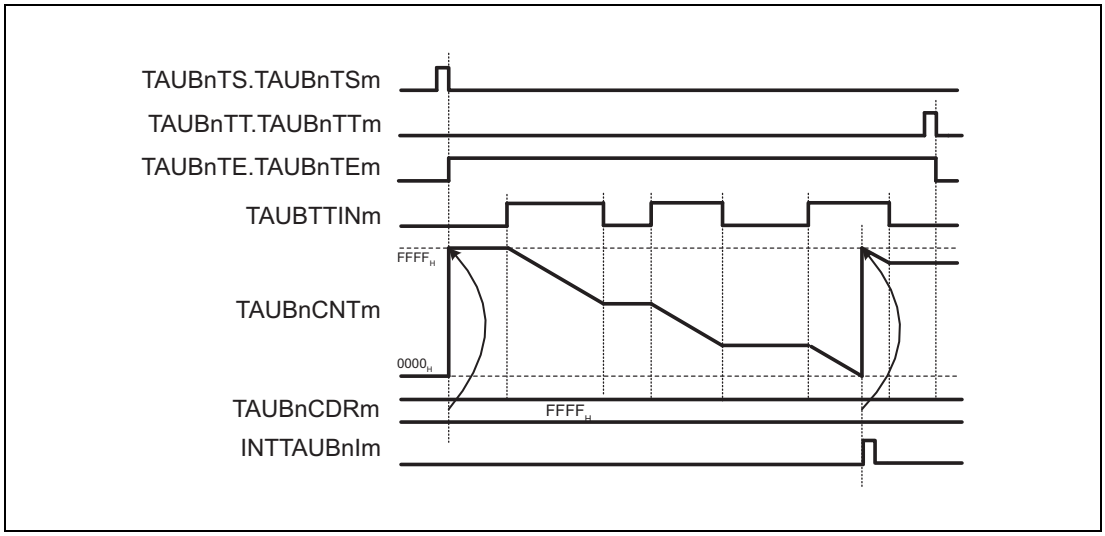



**Figure 24.75 General Timing Diagram for Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection)**

### 24.12.13.3 Register Settings

#### (1) TAUBnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.90 Contents of the TAUBnCMORm Register for Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $010_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $1100_B$. |
| 0 | TAUBnMD0 | Write $0_B$. |

#### (2) TAUBnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.91 Contents of the TAUBnCMURm Register for Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 10: Rising and falling edge detection (Low width measurement)<br>11: Rising and falling edge detection (High width measurement) |

### (3) Channel output mode

Set TAUBnTOE.TAUBnTOEm to 0 because the channel output mode is not used by this function.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUBnRDE, TAUBnRDS, TAUBnRDM, TAUBnRDC) cannot be used with the Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection). Therefore, these registers must be set to 0.

**Table 24.92 Simultaneous Rewrite Settings for Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection)**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 0: Disables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: When simultaneous rewrite is disabled (TAUBnRDE.TAUBnRDEm = 0), set these bits to 0 |
| TAUBnRDM.TAUBnRDMm | |
| TAUBnRDC.TAUBnRDCm | |

### 24.12.13.4 Operating Procedure for Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection)

Table 24.93 Operating Procedure for Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection)

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Set the TAUBnCMORm and TAUBnCMURm registers as described in **Table 24.90, Contents of the TAUBnCMORm Register for Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection)** and **Table 24.91, Contents of the TAUBnCMURm Register for Overflow Interrupt Output Function (during TAUBTTINm Input Period Count Detection)**<br><br>Set the value of the TAUBnCDRm register to $FFFF_H$. | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm to 1<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0.<br><br>Detection of TAUBTTINm start edge | TAUBnTE.TAUBnTEm is set to 1. TAUBnCNTm waits for detection of the start edge.<br><br>When a start edge is detected, TAUBnCNTm loads the TAUBnCDRm value ($FFFF_H$). |
| During operation | The TAUBnCNTm register can be read at all times. | TAUBnCNTm counts down. When the counter reaches $0000_H$:<br>• INTTAUBnIm is generated<br>• TAUBnCNTm reloads the TAUBnCDRm value ($FFFF_H$) and continues count operation<br>When a TAUBTTINm input stop edge is detected during count operation:<br>• TAUBnCNTm stops and retains its current value.<br>When a TAUBTTINm input start edge is detected while the counter is stopped:<br>• TAUBnCNTm starts to count down from the stop value.<br>Afterwards, this procedure is repeated. |
| Stop operation | Set TAUBnTT.TAUBnTTm to 1.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm stops and retains its current value. |

Restart operation

# 24.13 Independent Channel Simultaneous Rewrite Functions

The following describes functions that carry out simultaneous rewrite:

## 24.13.1 Simultaneous Rewrite Trigger Generation Function Type 1

### 24.13.1.1 Overview

#### Summary

This function generates an interrupt on a specific channel that can be used by lower channels as a simultaneous rewrite trigger. The interrupt is generated at regular intervals. The upper channel is for generating the simultaneous rewrite trigger (TAUBnRDC.TAUBnRDCm = 1), and the lower channels are for conducting simultaneous rewrite when triggered from the upper channel (TAUBnRDC.TAUBnRDCm = 0).

#### Prerequisites

- Two (or more) channels that are lower than the channel used as the upper channel, each with simultaneous rewrite enabled (TAUBnRDE.TAUBnRDEm = 1)
- The operation mode of the upper channel must be set to interval timer mode, see **Table 24.94, Contents of the TAUBnCMORm Register for the Upper Channel of the Simultaneous Rewrite Trigger Generation Function Type 1.**
- For the operation modes that can be set to the lower channels, see **Table 24.37, Channel Functions and the Methods They Use for Simultaneous Rewrite.**
- In this function, TAUBTTOUTm is not used for all the channels.

#### Description

The counters are enabled by setting the channel trigger bits (TAUBnTS.TAUBnTSm) of the upper and lower channel(s) to 1. This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation. The current value of the data register buffer of the upper channel (TAUBnCDRm buf) is written to the counter (TAUBnCNTm) and the counter starts to count down from this value.
The counter(s) of the lower channel(s) start to count as specified by their selected operating modes.

When a counter reaches $0000_H$, an interrupt is generated from the channel.
The corresponding TAUBnCNTm then reloads the current TAUBnCDRm buffer value and subsequently continues operation.

If the channel where the interrupt occurs is specified as the trigger channel for simultaneous rewrite (TAUBnRDC.TAUBnRDCm = 1) and is an upper channel, simultaneous rewrite takes place on all lower channels in which simultaneous rewrite is currently possible (TAUBnRSF.TAUBnRSFm = 1).

The values of the data registers are copied to the corresponding data register buffers. Each time a counter starts to count down, it reads the value in the data register buffer and counts down from this value.

The value of a data register can be changed at any time, but it is only transferred to the corresponding data register buffer when simultaneous rewrite occurs.

**Condition**

- The channel which is monitored for INTTAUBnIm is specified by setting TAUBnRDC.TAUBnRDCm = 1 for the corresponding channel. The TAUBnRDC.TAUBnRDCm bit must be 0 for all other channels in which simultaneous rewrite should take place.

#### 24.13.1.2 Equations

Simultaneous rewrite trigger generation cycle = count clock cycle × (TAUBnCDRm + 1)

To control simultaneous rewrite, the following condition must be satisfied:

**[For PWM]**

TAUBnCDRm = [(value of TAUBnCDRm of master channel subject to simultaneous rewrite + 1) × number of interrupts] – 1

**[For triangle PWM]**

TAUBnCDRm = [(value of TAUBnCDRm of master channel subject to simultaneous rewrite + 1) × 2 × number of interrupts] – 1

That is, the ratio of TAUBnCDRm + 1 and value of TAUBnCDRm of master channel subject to simultaneous rewrite + 1 must be an integer. This integer corresponds to the number of interrupts.

Note that the cycle for the triangle PWM is twice the cycle for the PWM

### 24.13.1.3 Block Diagram and General Timing Diagram

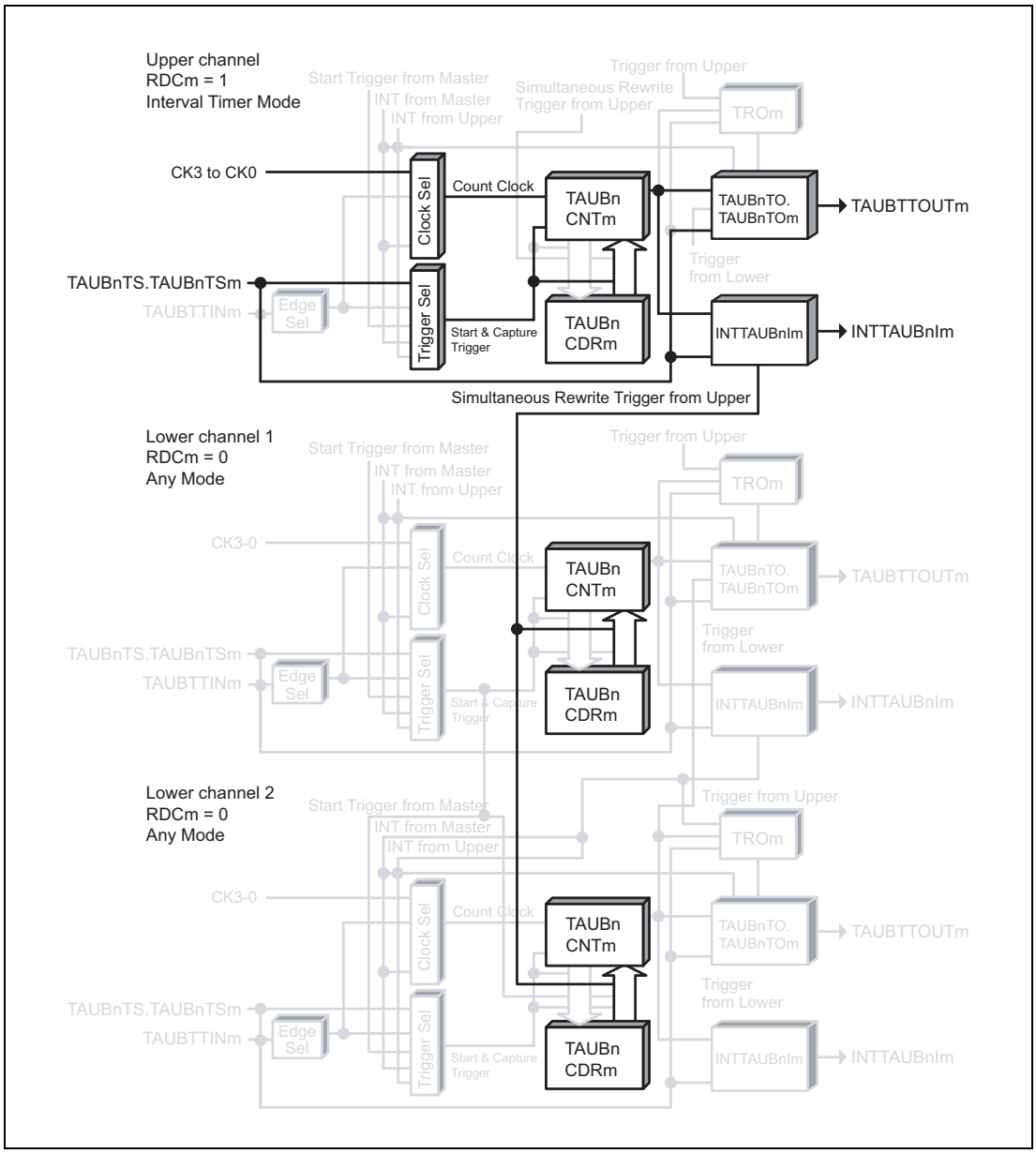



**Figure 24.76 Block Diagram for Simultaneous Rewrite Trigger Generation Function Type 1**

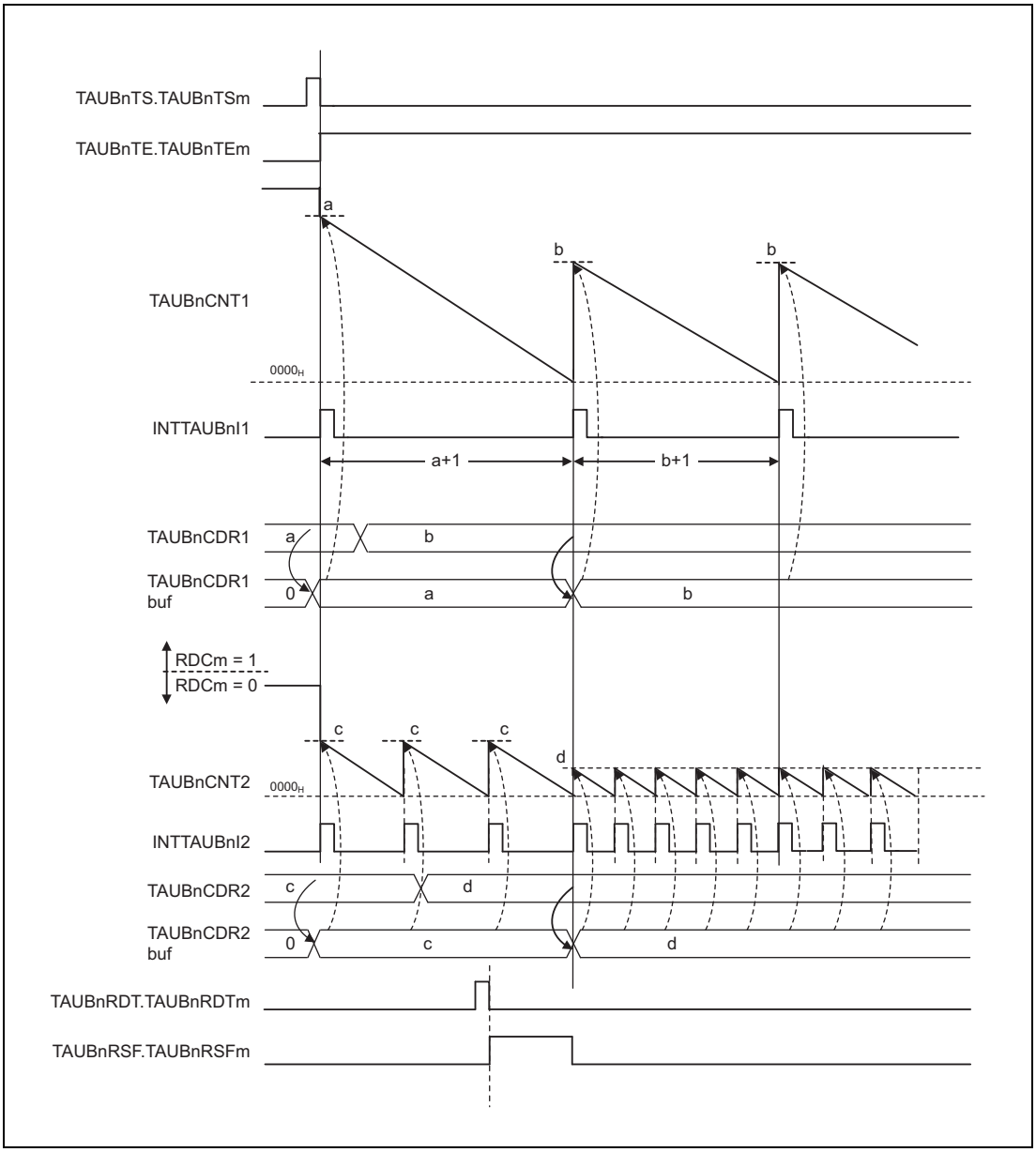



**Figure 24.77 General Timing Diagram for Simultaneous Rewrite Trigger Generation Function Type 1**

### 24.13.1.4 Register Settings for The Upper Channel

#### (1) TAUBnCMORm for the upper channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.94 Contents of the TAUBnCMORm Register for the Upper Channel of the Simultaneous Rewrite Trigger Generation Function Type 1**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $000_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0000_B$. |
| 0 | TAUBnMD0 | Write $1_B$. |

#### (2) TAUBnCMURm for the upper channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.95 Contents of the TAUBnCMURm Register for the Upper Channel of the Simultaneous Rewrite Trigger Generation Function Type 1**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Not used, so set to 00. |

#### (3) Channel output mode for the upper channel

Set TAUBnTOE.TAUBnTOEm to 0 because the channel output mode is not used by this function. However, it can be used in independent channel output mode controlled by software.

#### (4) Simultaneous rewrite for the upper channel

Table 24.96 Simultaneous Rewrite Settings for the Upper Channel in Simultaneous Rewrite Trigger Generation Function Type 1

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 1: Enables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 1: Selects an upper channel as the control channel for simultaneous rewrite |
| TAUBnRDM.TAUBnRDMm | 0: The signal that controls simultaneous rewrite is loaded when the master channel starts counting |
| TAUBnRDC.TAUBnRDCm | 1: Channel is monitored for an INTTAUBnIm signal that is used as the simultaneous rewrite trigger |

### 24.13.1.5 Register Settings for the Lower Channel(s)

#### (1) TAUBnCMORm for the lower channel(s)

For the TAUBnCMORm register of the lower channels, follow the TAUBnCMORm register settings for the operation mode that can be set. (See **Table 24.37, Channel Functions and the Methods They Use for Simultaneous Rewrite**)

#### (2) TAUBnCMURm for the lower channel(s)

For the TAUBnCMURm register of the lower channels, follow the TAUBnCMURm register settings for the operation mode that can be set. (See **Table 24.37, Channel Functions and the Methods They Use for Simultaneous Rewrite**)

#### (3) Channel output mode for the lower channel(s)

Output can be made according to the setting for lower channels (master/slave). As for the available function for simultaneous rewrite trigger generation function type 1, see **Table 24.37, Channel Functions and the Methods They Use for Simultaneous Rewrite**.

#### (4) Simultaneous rewrite for the lower channel(s)

Table 24.97 Simultaneous Rewrite Settings for the Lower Channel in Simultaneous Rewrite Trigger Generation Function Type 1

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 1: Enables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 1: Selects an upper channel as the control channel for simultaneous rewrite |
| TAUBnRDM.TAUBnRDMm | 0: The signal that controls simultaneous rewrite is loaded when the master channel starts counting |
| TAUBnRDC.TAUBnRDCm | 0: Does not use the channel to generate the simultaneous rewrite trigger. |

### 24.13.1.6 Operating Procedure for Simultaneous Rewrite Trigger Generation Function Type 1

**Table 24.98 Operating Procedure for Simultaneous Rewrite Trigger Generation Function Type 1**

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Set the TAUBnCMORm and TAUBnCMURm registers for the upper channel as described in **Table 24.94, Contents of the TAUBnCMORm Register for the Upper Channel of the Simultaneous Rewrite Trigger Generation Function Type 1** and **Table 24.95, Contents of the TAUBnCMURm Register for the Upper Channel of the Simultaneous Rewrite Trigger Generation Function Type 1**<br><br>Set the TAUBnCMORm and TAUBnCMURm registers for the lower channel as described in **Section 24.13.1.5, Register Settings for the Lower Channel(s)**<br><br>Set the value of the TAUBnCDRm register | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm to 1.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is set to 1 and the counter starts.<br>TAUBnCNTm loads the TAUBnCDRm value.<br>When TAUBnCMORm.TAUBnMD0 = 1, INTTAUBnIm is generated. |
| During operation | TAUBnRDT.TAUBnRDTm, TAUBnCDRm can be changed.<br>TAUBnRSF.TAUBnRSFm can be read at all times. | TAUBnCNTm counts down. When the counter reaches $0000_H$:<br>• TAUBnCNTm reloads the TAUBnCDRm value and continues count operation<br>• INTTAUBnIm is generated<br>Simultaneous rewrite is controlled when INTTAUBnIm is generated from the channel where TAUBnRDC.TAUBnRDCm is set to 1. Afterwards, this procedure is repeated. |
| Stop operation | Set TAUBnTT.TAUBnTTm to 1.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm stops and retains its current value. |

Restart operation

# 24.14 Synchronous Channel Operation Functions

This section lists all the synchronous channel operation functions provided by the Timer Array Unit B. For a general overview of synchronous channel operation, see **Section 24.2, Overview**.

## 24.14.1 PWM Output Function

### 24.14.1.1 Overview

#### Summary

This function generates multiple PWM outputs by using a master and multiple slave channels. It enables the pulse cycle (frequency) and the pulse width (duration) of the TAUBTTOUTm to be set. The pulse cycle is set in the master channel. The pulse width is set in the slave channel.

#### Prerequisites

- Two channels
- The operation mode of the master channel must be set to interval timer mode, see **Table 24.99, Contents of the TAUBnCMORm Register for the Master Channel of the PWM Output Function**
- The operation mode of the slave channel(s) must be set to one-count mode, see **Table 24.102, Contents of the TAUBnCMORm Register for the Slave Channel of the PWM Output Function**
- TAUBTTOUTm is not used for the master channel of this function
- The channel output mode of the slave channel(s) must be set to synchronous channel output mode 1.

#### Description

The counters are enabled by setting the channel trigger bits (TAUBnTS.TAUBnTSm) to 1. This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation. The current value of TAUBnCDRm is written to TAUBnCNTm and the counters start to count down from these values. INTTAUBnIm is generated on the master channel and TAUBTTOUTm (slave) toggles, which realizes a PWM output.

- Master channel:

  When the counter of the master channel reaches $0000_H$, pulse cycle time has elapsed and INTTAUBnIm is generated. The counter loads the TAUBnCDRm value and counts down.

- Slave channel:

  INTTAUBnIm generated on the master channel triggers the counter of the slave channel(s). The current value of TAUBnCDRm (slave) is written to TAUBnCNTm (slave) and the counter starts to count down from this value. The TAUBTTOUTm signal is set to the active level.

  When the counter reaches $0000_H$, i.e. duty time has elapsed, INTTAUBnIm is generated and the TAUBTTOUTm signal is reset to the inactive level. The counter returns to $FFFF_H$ and awaits the next INTTAUBnIm of the master channel, and thus the start of the next pulse cycle.

The counter can be stopped by setting TAUBnTT.TAUBnTTm to 1 for the master and slave channel(s), which in turn sets TAUBnTE.TAUBnTEm to 0. TAUBnCNTm and TAUBTTOUTm of master and slave channel(s) stop but retain their values. The counters can be restarted by setting TAUBnTS.TAUBnTSm to 1.

**Conditions**

Simultaneous rewrite can be used with this function. Please see **Section 24.6, Simultaneous Rewrite**.

### 24.14.1.2 Equations

Pulse cycle = (TAUBnCDRm (master) + 1) × count clock cycle

Duty cycle [%] = (TAUBnCDRm (slave) / (TAUBnCDRm (master) + 1)) × 100

– Duty cycle = 0%
  TAUBnCDRm (slave) = $0000_H$
– Duty cycle = 100%
  TAUBnCDRm (slave) ≥ TAUBnCDRm (master) + 1

### 24.14.1.3 Block Diagram and General Timing Diagram

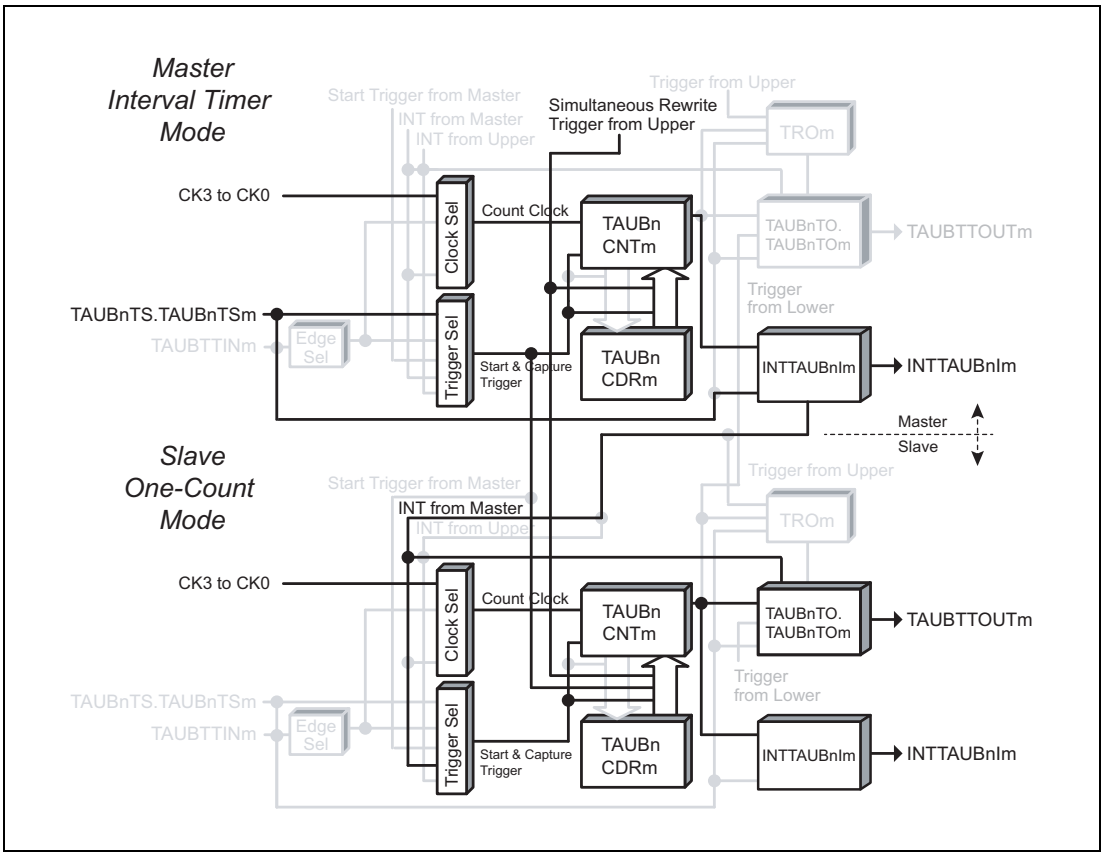



**Figure 24.78 Block Diagram for PWM Output Function**

The following settings apply to the general timing diagram.

- Slave channel: Positive logic (TAUBnTOL.TAUBnTOLm = 0)

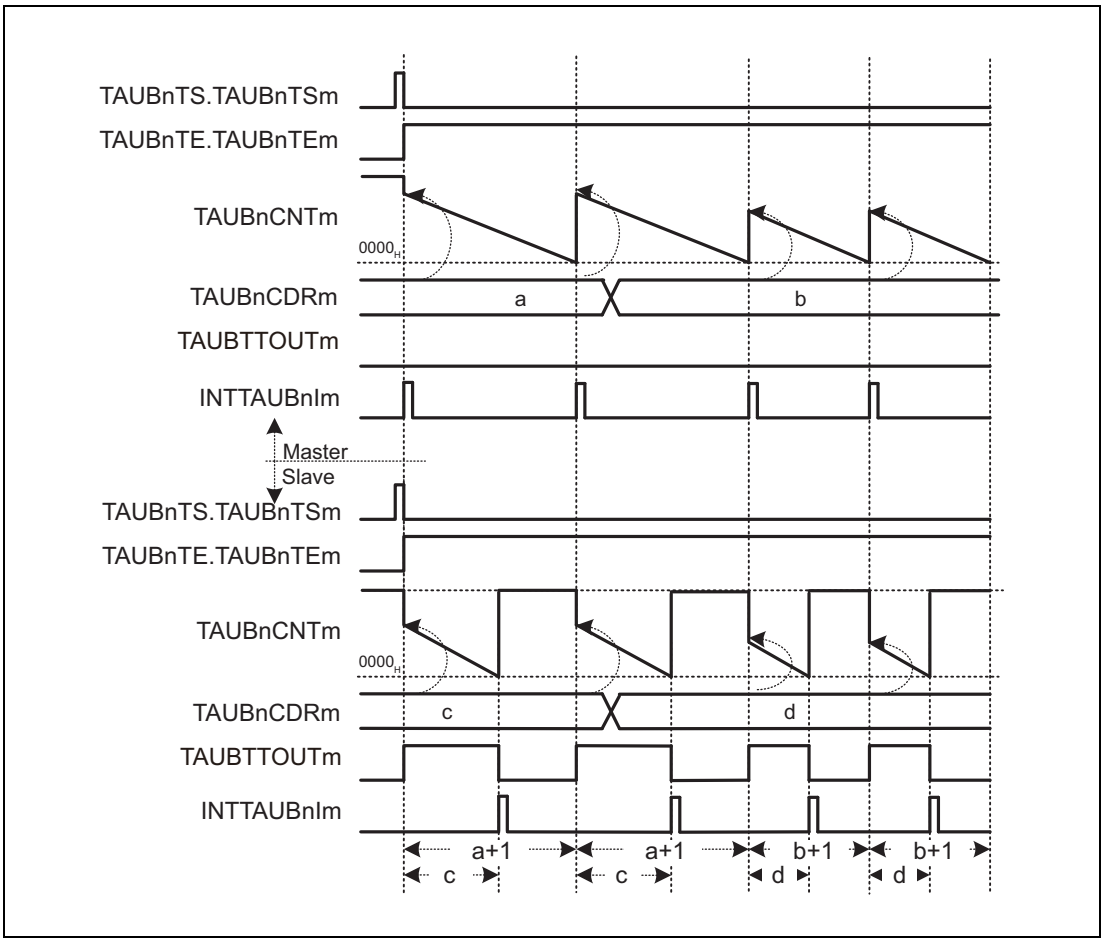



**Figure 24.79 General Timing Diagram for PWM Output Function**

**NOTE**

The interval between the starting to count and an interrupt being generated is the value of corresponding TAUBnCDRm + 1.

### 24.14.1.4 Register Settings for the Master Channel

#### (1) TAUBnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.99 Contents of the TAUBnCMORm Register for the Master Channel of the PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUBnCKS[1:0] bits of the master and slave channel(s) must be identical. |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $1_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $000_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0000_B$. |
| 0 | TAUBnMD0 | Write $1_B$. |

#### (2) TAUBnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.100 Contents of the TAUBnCMURm Register for the Master Channel of the PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Not used, so set to 00. |

### (3) Channel output mode for the master channel

The channel output mode is not used by this function.

### (4) Simultaneous rewrite for the master channel

The simultaneous rewrite settings of the master and slave channels must be identical.

**Table 24.101 Simultaneous Rewrite Settings for the Master Channel of the PWM Output Function**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 1: Enables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: Selects master channel for simultaneous rewrite triggers.<br>1: Selects upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUBnRDM.TAUBnRDMm | 0: The simultaneous rewrite trigger signal is generated when the master channel starts counting |
| TAUBnRDC.TAUBnRDCm | 0: Does not use the channel to generate the simultaneous rewrite trigger. |

**NOTE**

When used in TAUBnRDS.TAUBnRDSm = 1, the master channel requires an upper channel operating in **Section 24.13.1, Simultaneous Rewrite Trigger Generation Function Type 1**.

Configure the operation following the conditions below.

- The channel set to Simultaneous Rewrite Trigger Output Function Type 1: TAUBnRDCm = 1, TAUBnRDSm = 1
  The setting value of TAUBnCDRm to this channel is as follows.
  = ((setting value of TAUBnCDRm of the master channel subject to simultaneous rewrite + 1) × number of interrupts) – 1
- Master channel: TAUBnRDCm = 0, TAUBnRDSm = 1
- Slave channel: TAUBnRDCm = 0, TAUBnRDSm = 1

Although the value of duty exceeds 100% when the setting value of TAUBnCDRm (slave) > the setting value of TAUBnCDRm (master) + 1, the output will be aggregated to 100%.

### 24.14.1.5 Register Settings for the Slave Channel(s)

#### (1) TAUBnCMORm for the slave channel(s)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.102 Contents of the TAUBnCMORm Register for the Slave Channel of the PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUBnCKS[1:0] bits of the master and slave channel(s) must be identical. |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $100_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0100_B$. |
| 0 | TAUBnMD0 | Write $1_B$. |

#### (2) TAUBnCMURm for the slave channel(s)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.103 Contents of the TAUBnCMURm Register for the Slave Channel of the PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Not used, so set to 00. |

### (3) Channel output mode for the slave channel(s)

**Table 24.104 Control Bit Settings for Synchronous Channel Output Mode 1**

| Bit Name | Setting |
|---|---|
| TAUBnTOE.TAUBnTOEm | Write $1_B$. |
| TAUBnTOM.TAUBnTOMm | Write $1_B$. |
| TAUBnTOC.TAUBnTOCm | Write $0_B$. |
| TAUBnTOL.TAUBnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUBnTDE.TAUBnTDEm | Write $0_B$. |
| TAUBnTDL.TAUBnTDLm | Write $0_B$. |

### (4) Simultaneous rewrite for the slave channel(s)

The simultaneous rewrite settings of the master and slave channels must be identical.

**Table 24.105 Simultaneous Rewrite Settings for the Slave Channel of the PWM Output Function**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 1: Enables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: Selects master channel for simultaneous rewrite triggers.<br>1: Selects upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUBnRDM.TAUBnRDMm | 0: The simultaneous rewrite trigger signal is generated when the master channel starts counting |
| TAUBnRDC.TAUBnRDCm | 0: Does not use the channel to generate the simultaneous rewrite trigger. |

### 24.14.1.6 Operating Procedure for PWM Output Function

**Table 24.106 Operating Procedure for PWM Output Function**

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Master channel: set the TAUBnCMORm and TAUBnCMURm registers and the channel output mode as described in **Section 24.14.1.4, Register Settings for the Master Channel**.<br><br>Slave channel: set the TAUBnCMORm and TAUBnCMURm registers and the channel output mode as described in **Section 24.14.1.5, Register Settings for the Slave Channel(s)**.<br><br>Set the values of the TAUBnCDRm registers of all channels | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm of the master and slave channels to 1 simultaneously.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm (master and slave channels) is set to 1 and the counters of the master and slave channels start.<br>INTTAUBnIm is generated on the master channel and TAUBTTOUTm (slave) is set. |
| During operation | TAUBnCDRm can be changed at any time.<br>TAUBnTOL.TAUBnTOLm can be changed.<br>TAUBnCNTm and TAUBnRSF.TAUBnRSFm can be read at any time.<br><br>TAUBnRDT.TAUBnRDTm can be changed during operation. | TAUBnCNTm of the master channel loads TAUBnCDRm and counts down. When the counter reaches $0000_H$:<br>• INTTAUBnIm (master) is generated<br>• TAUBnCNTm (master) loads the TAUBnCDRm value and continues count operation<br>• TAUBnCNTm (slave) loads the TAUBnCDRm value and counts down<br>• TAUBTTOUTm (slave) is set to the active level<br>When TAUBnCNTm (slave) reaches $0000_H$:<br>• The counter of TAUBnCNTm (slave) stops.<br>• INTTAUBnIm (slave) is generated<br>• TAUBTTOUTm (slave) is set to the inactive level |
| Stop operation | Set TAUBnTT.TAUBnTTm of the master and slave channels to 1 simultaneously.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm and TAUBTTOUTm stop and retain their current values. |

Restart operation

### 24.14.1.7 Specific Timing Diagrams

#### (1) Duty cycle = 0%

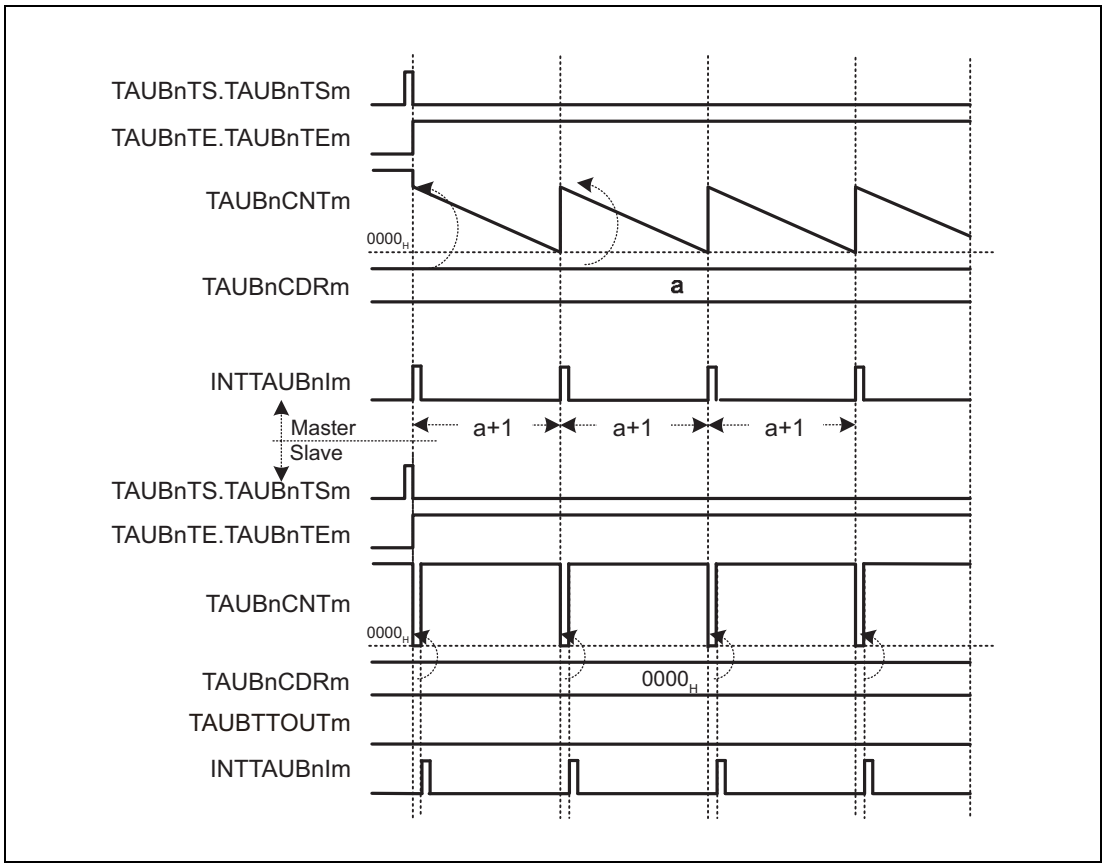



**Figure 24.80 TAUBnCDRm (slave) = $0000_H$, Positive Logic (TAUBnTOL.TAUBnTOLm (slave) = 0)**

- Every time the master channel generates an interrupt (INTTAUBnIm), $0000_H$ is written to TAUBnCNTm (slave). As a result, a slave channel interrupt (INTTAUBnIm) is generated at the same time and TAUBTTOUTm remains inactive.
- TAUBnCNTm (slave) generates an interrupt every time the value of TAUBnCDRm is loaded.

**(2) Duty cycle = 100%**

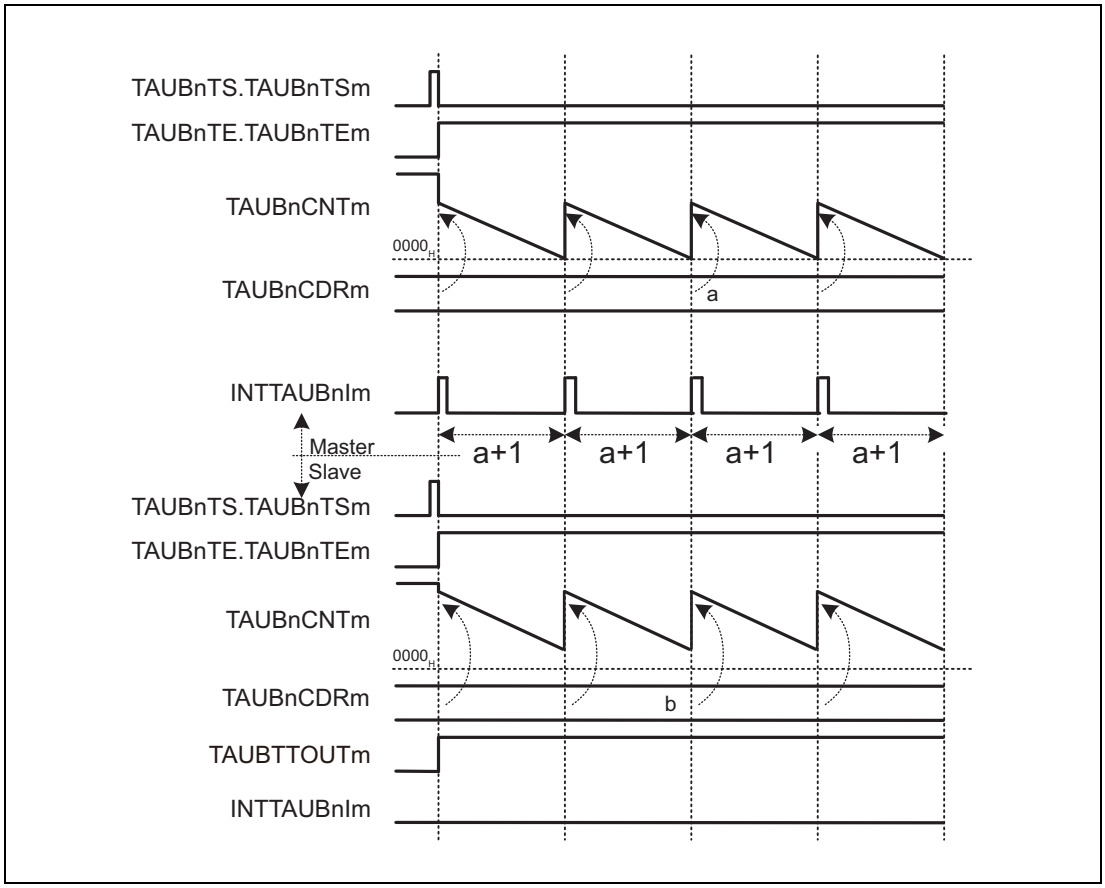



**Figure 24.81 TAUBnCDRm (slave) ≥ TAUBnCDRm (master) + 1, Positive Logic (TAUBnTOL.TAUBnTOLm (slave) = 0)**

If the value TAUBnCDRm (slave) is higher than the value TAUBnCDRm (master), the counter of the slave channel cannot reach $0000_H$ and cannot generate interrupts. The TAUBTTOUTm remains at active state.

### (3) Operation stop and restart

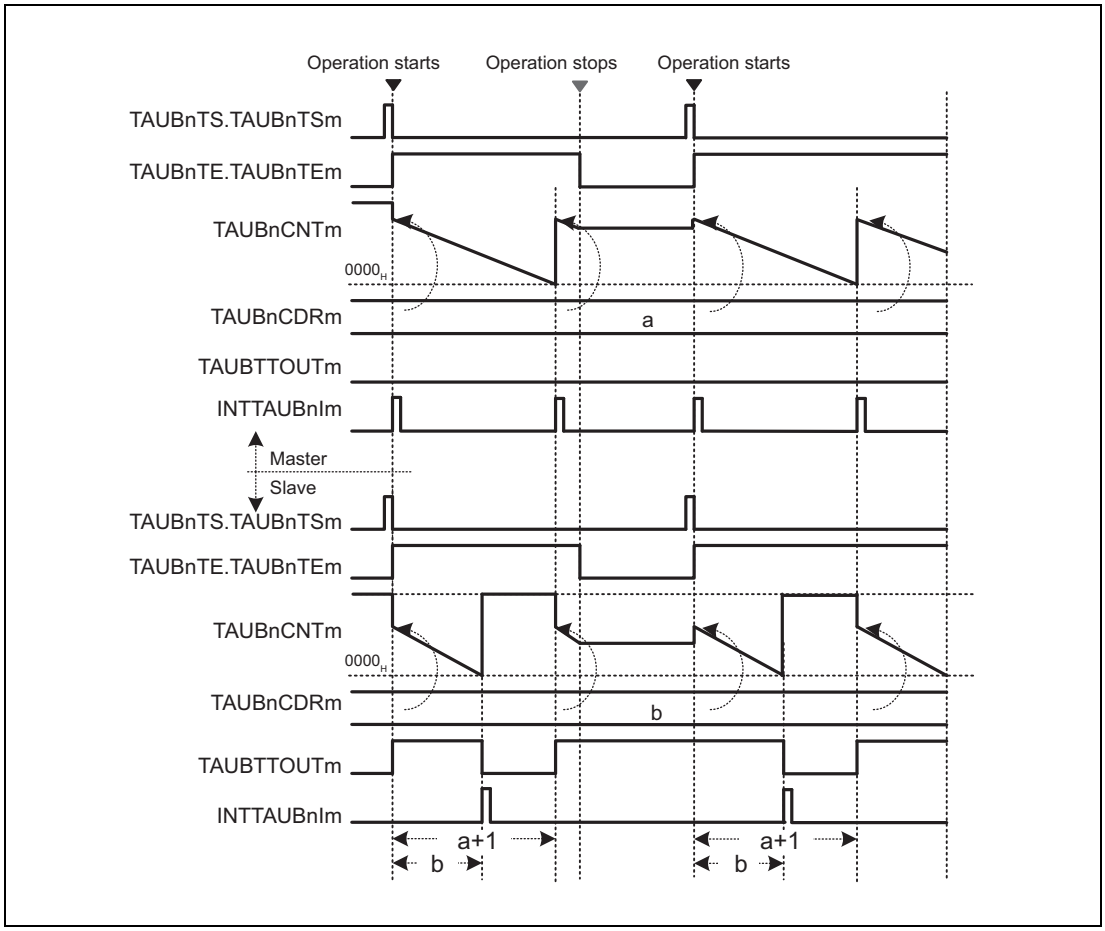



**Figure 24.82 Operation Stop and Restart, Positive Logic (TAUBnTOL.TAUBnTOLm (slave) = 0)**

- The counter can be stopped by setting TAUBnTT.TAUBnTTm of the master and slave channel(s) to 1, which in turn sets TAUBnTE.TAUBnTEm to 0.
- TAUBnCNTm and TAUBTTOUTm of all channels stop and the current values are retained. No interrupts are generated.
- The counter can be restarted by setting TAUBnTS.TAUBnTSm of master and slave channel(s) to 1. TAUBnCNTm of master and slave channel reload the current values of TAUBnCDRm and start to count down from these values.

### (4) Operation stop and restart (Slave output, Initialization)

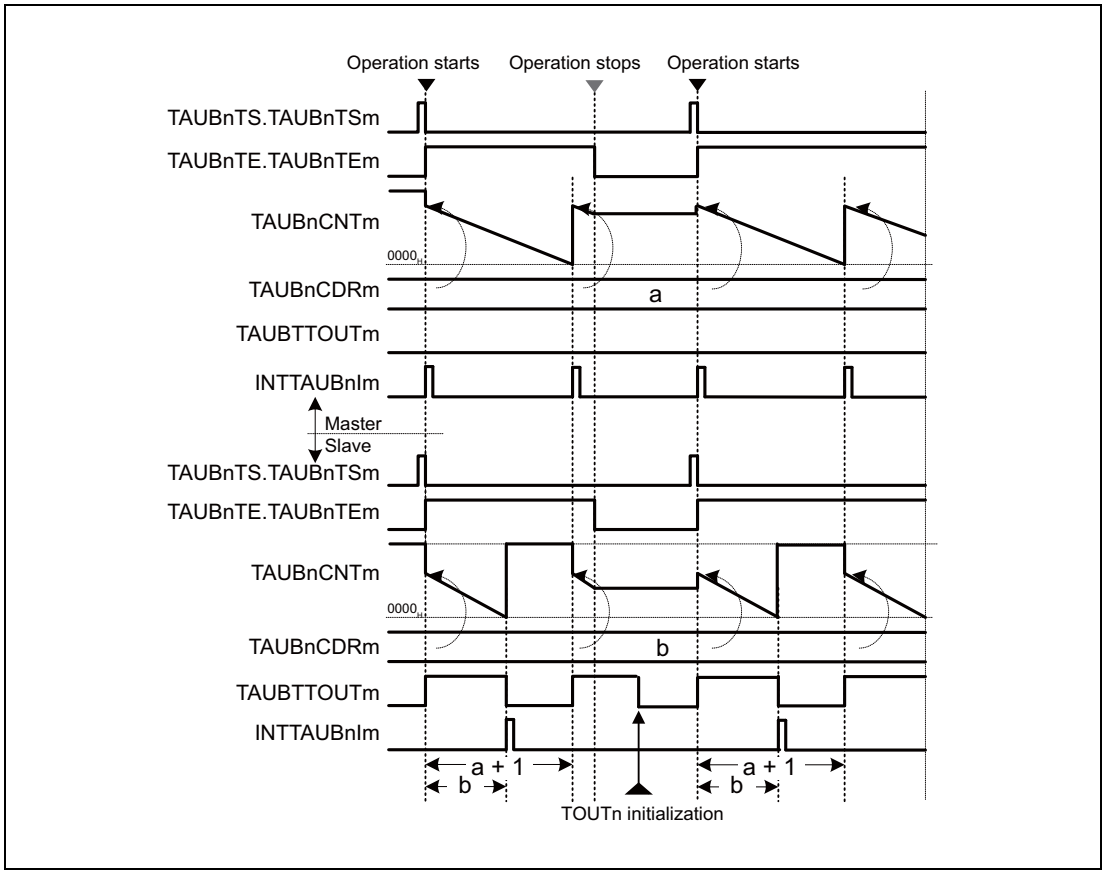



**Figure 24.83 Operation Stop and Restart (Slave Output, Initialization)**

When TAUBnTOE.TAUBnTOEm of the slave channel is set to 0 while TAUBnTE.TAUBnTEm = 0 and the inactive level of TAUBTTOUTm is written in the TAUBnTO.TAUBnTOm, the output level of TAUBTTOUTm (slave channel) becomes active when INTTAUBnIm is issued when the count operation is started after restart.

## 24.14.2 One-Shot Pulse Output Function

### 24.14.2.1 Overview

#### Summary

This function outputs a signal pulse with a defined pulse width and a specific delay time compared to an external input signal pulse by using a master and a slave channel. The delay time is specified using the master channel. The pulse width is specified using the slave channel.

#### Prerequisites

- Two channels
- The operation mode of the master channel must be set to one-count mode, see **Table 24.107, Contents of the TAUBnCMORm Register for the Master Channel of the One-Shot Pulse Output Function**
- The operation mode of the slave channel must be set to pulse one-count mode, see **Table 24.110, Contents of the TAUBnCMORm Register for the Slave Channel of the One-Shot Pulse Output Function**
- TAUBTTOUTm is not used for the master channel of this function
- The channel output mode of the slave channel must be set to independent channel output mode 2.
- TAUBTTINm (master) has to be detected while TAUBnCNTm (master) and TAUBnCNTm (slave) await a trigger. Furthermore, the slave is only triggered by an interrupt from the master channel and not by TAUBTTINm (slave).

#### Description

The counters are enabled by setting the channel trigger bits (TAUBnTS.TAUBnTSm) for master and slave channels to 1. This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation.

- Master channel:
  When the next valid TAUBTTINm input edge is detected, the current value of TAUBnCDRm is written to TAUBnCNTm. The counter starts to count down from this value. If TAUBnCMORm.TAUBnMD0 = 0, a trigger (TAUBTTINm) which is detected within the delay time is ignored.
  When the counter of the master channel reaches $0000_H$, INTTAUBnIm is generated. The counter returns to $FFFF_H$ and awaits the next valid TAUBTTINm input edge.
- Slave channel:
  INTTAUBnIm generated on the master channel triggers the counter of the slave channel. The current value of TAUBnCDRm (slave) is written to TAUBnCNTm (slave) and the counter starts to count down from this value.
  An interrupt is generated and the TAUBTTOUTm signal is set.
  When the counter reaches $0001_H$, INTTAUBnIm is generated and the TAUBTTOUTm signal is reset. The counter remains at $0000_H$ and awaits the next INTTAUBnIm of the master channel.

The counter can be stopped by setting TAUBnTT.TAUBnTTm to 1 for the master and slave channels, which in turn sets TAUBnTE.TAUBnTEm to 0. TAUBnCNTm and TAUBTTOUTm of master and slave channels stop but retain their values. The counters can be restarted by setting TAUBnTS.TAUBnTSm to 1.

The counter of the master channel can be restarted without stopping it first (forced restart) by setting TAUBnTS.TAUBnTSm to 1 during operation.

**NOTES**

1. If a forced restart of the counter is executed during operation, the width of the output signal does not correspond to the value of TAUBnCDRm (slave).
2. The TAUBTTINm input signal is sampled at the frequency of the operating clock, specified by TAUBnCMORm.TAUBnCKS[1:0] bits. As a result, the output cycle of TAUBTTOUTm has an error of ± 1 operation clock cycle.

**Conditions**

- If TAUBnCMORm.TAUBnMD0 of the master channel is set to 0, during counting detected TAUBTTINm input edges are ignored.
- Simultaneous rewrite can be used with this function. Please see **Section 24.6, Simultaneous Rewrite**.

#### 24.14.2.2 Equations

Delay from trigger input to pulse output
= (TAUBnCDRm (master) + 1) × count clock cycle

Pulse width = (TAUBnCDRm (slave)) × count clock cycle

### 24.14.2.3 Block Diagram and General Timing Diagram

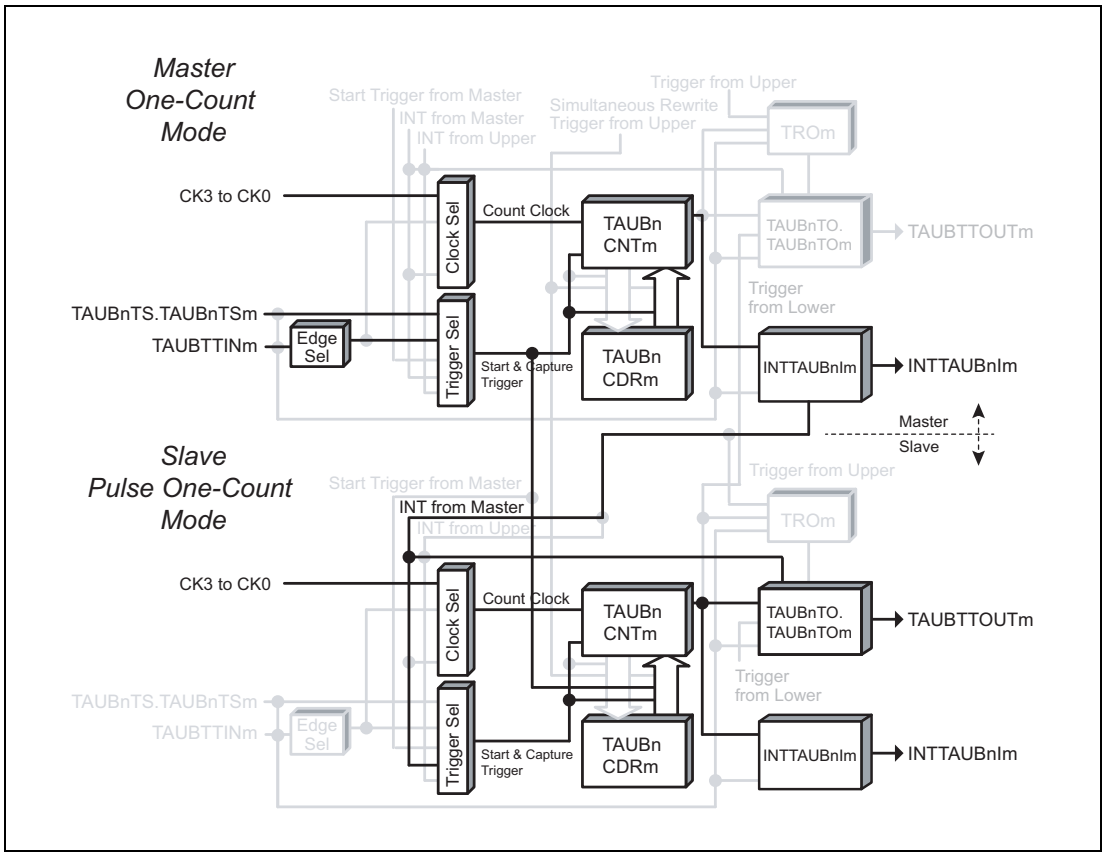



**Figure 24.84 Block Diagram for One-Shot Pulse Output Function**

The following settings apply to the general basic diagram.

- Start trigger detection disabled during counting (TAUBnCMORm.TAUBnMD0 = 0)
- Falling edge detection (TAUBnCMURm.TAUBnTIS[1:0] = $00_B$)

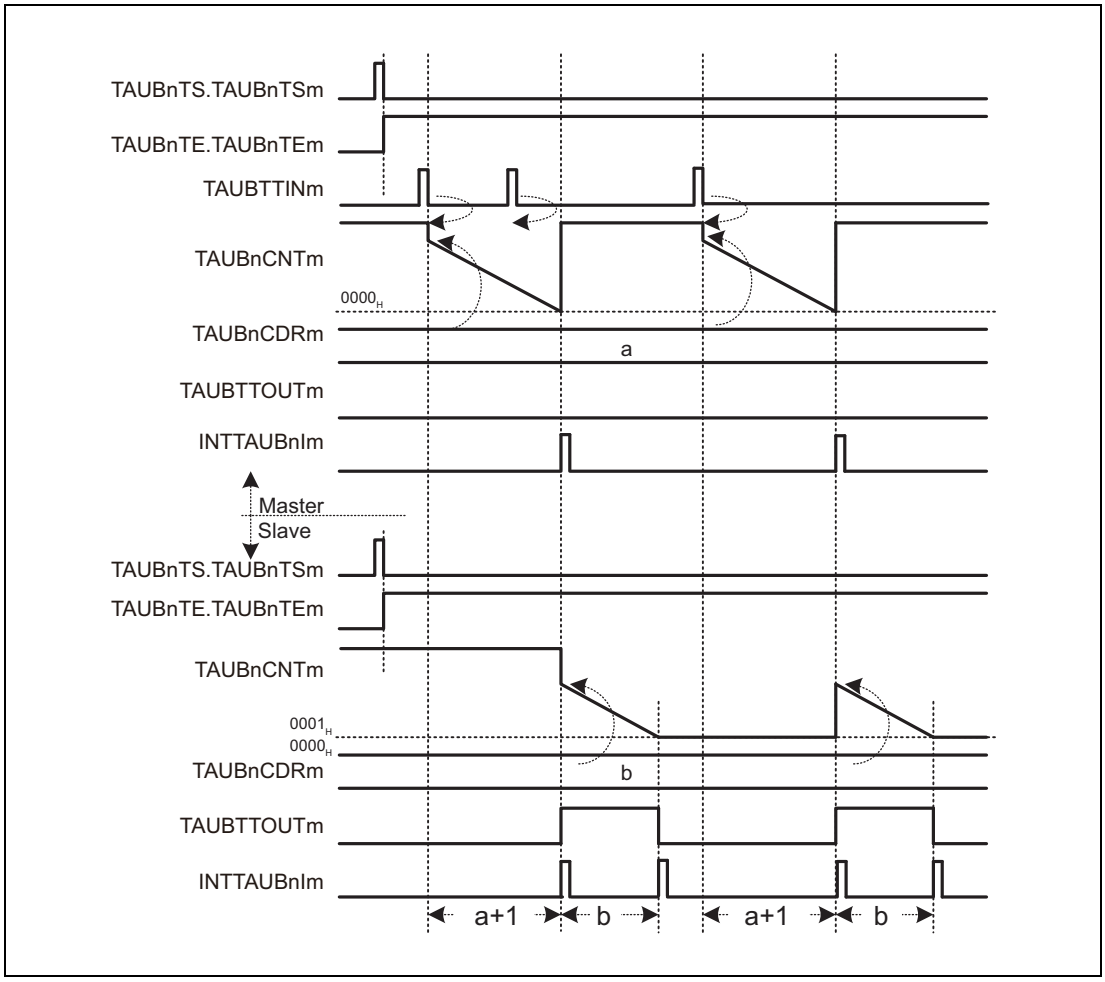



**Figure 24.85 General Timing Diagram for One-Shot Pulse Output Function**

### 24.14.2.4 Register Settings for the Master Channel

#### (1) TAUBnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.107 Contents of the TAUBnCMORm Register for the Master Channel of the One-Shot Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUBnCKS[1:0] bits of the master and slave channel(s) must be identical. |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $1_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $001_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0100_B$. |
| 0 | TAUBnMD0 | 0: Disables start trigger detection during counting<br>1: Enables start trigger detection during counting |

#### (2) TAUBnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.108 Contents of the TAUBnCMURm Register for the Master Channel of the One-Shot Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Falling edge detection<br>01: Rising edge detection<br>10: Rising and falling edge detection<br>11: Setting prohibited |

### (3) Channel output mode for the master channel

Set TAUBnTOE.TAUBnTOEm to 0 because the channel output mode is not used by this function.

### (4) Simultaneous rewrite for the master channel

The simultaneous rewrite settings of the master and slave channels must be identical.

**Table 24.109 Simultaneous Rewrite Settings for the Master Channel of the One-Shot Pulse Output Function**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 1: Enables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: The master channel is the control channel for simultaneous rewrite |
| TAUBnRDM.TAUBnRDMm | 0: The simultaneous rewrite trigger signal is generated when the master channel starts counting |
| TAUBnRDC.TAUBnRDCm | 0: Does not use the channel to generate the simultaneous rewrite trigger. |

### 24.14.2.5 Register Settings for the Slave Channel

#### (1) TAUBnCMORm for the slave channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.110 Contents of the TAUBnCMORm Register for the Slave Channel of the One-Shot Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUBnCKS[1:0] bits of the master and slave channels must be identical. |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $100_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $1010_B$. |
| 0 | TAUBnMD0 | 0: Disables start trigger detection during counting<br>1: Enables start trigger detection during counting<br>The value of the MD0 bit of the master and slave channels must be identical. |

#### (2) TAUBnCMURm for the slave channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.111 Contents of the TAUBnCMURm Register for the Slave Channel of the One-Shot Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Not used, so set to 00. |

### (3) Channel output mode for the slave channel

Table 24.112 Control Bit Settings for Independent Channel Output Mode 2

| Bit Name | Setting |
|---|---|
| TAUBnTOE.TAUBnTOEm | 1: Enables independent channel output mode |
| TAUBnTOM.TAUBnTOMm | 0: Independent channel output |
| TAUBnTOC.TAUBnTOCm | 1: Operating mode 2 |
| TAUBnTOL.TAUBnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUBnTDE.TAUBnTDEm | 0: Disables dead time operation |
| TAUBnTDL.TAUBnTDLm | 0: When dead time operation is disabled (TAUBnTDE.TAUBnTDEm = 0), set these bits to 0 |

### (4) Simultaneous rewrite for the slave channel

The simultaneous rewrite settings of the master and slave channels must be identical.

Table 24.113 Simultaneous Rewrite Settings for the Slave Channel of the One-Shot Pulse Output Function

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 1: Enables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: The master channel is the control channel for simultaneous rewrite |
| TAUBnRDM.TAUBnRDMm | 0: The simultaneous rewrite trigger signal is generated when the master channel starts counting |
| TAUBnRDC.TAUBnRDCm | 0: Does not use the channel to generate the simultaneous rewrite trigger. |

### 24.14.2.6 Operating Procedure for One-Shot Pulse Output Function

Table 24.114 Operating Procedure for One-Shot Pulse Output Function

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Master channel: set the TAUBnCMORm and TAUBnCMURm registers and the channel output mode as described in **Section 24.14.2.4, Register Settings for the Master Channel**.<br><br>Slave channel: set the TAUBnCMORm and TAUBnCMURm registers and the channel output mode as described in **Section 24.14.2.5, Register Settings for the Slave Channel**.<br><br>Set the values of the TAUBnCDRm registers of all channels | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm of the master and slave channels to 1 simultaneously.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm (master and slave channels) is set to 1 and the master channel awaits a TAUBTTINm input. |
| During operation | TAUBnCDRm can be changed at any time. TAUBnCNTm and TAUBnRSF.TAUBnRSFm can be read at any time.<br><br>TAUBnRDT.TAUBnRDTm can be changed during operation. | When a valid TAUBTTINm input edge is detected, TAUBnCNTm of the master channel loads the TAUBnCDRm value and counts down. When the counter reaches $0000_H$:<br>• INTTAUBnIm (master) is generated<br>• TAUBnCNTm (master) is reset to $FFFF_H$ and waits for the next valid TAUBTTINm input edge.<br>• TAUBnCNTm (slave) reloads the TAUBnCDRm value and starts to count down<br>• INTTAUBnIm (slave) is generated<br>• TAUBTTOUTm (slave) is set to the active level.<br>When TAUBnCNTm (slave) reaches $0001_H$:<br>• The counter of TAUBnCNTm (slave) stops.<br>• INTTAUBnIm (slave) is generated<br>• TAUBTTOUTm (slave) is set to an inactive level. |
| Stop operation | Set TAUBnTT.TAUBnTTm of the master and slave channels to 1 simultaneously.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm and TAUBTTOUTm stop and retain their current values. |

Restart operation

### 24.14.2.7 Specific Timing Diagrams

#### (1) TAUBnCDRm (master) = $0000_H$

The following settings apply to this diagram.

- Start trigger detection disabled during counting (TAUBnCMORm.TAUBnMD0 = 0)
- Falling edge detection (TAUBnCMURm.TAUBnTIS[1:0] = $00_B$)

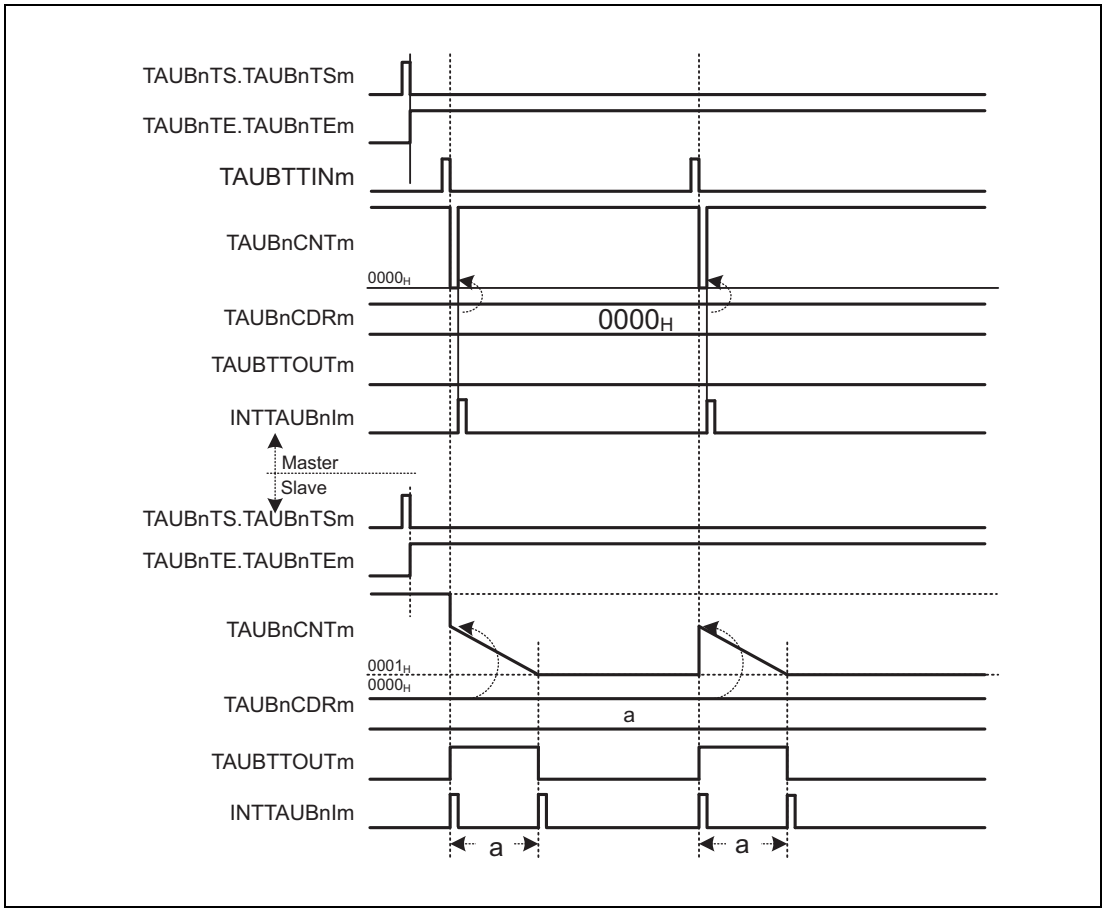



**Figure 24.86 TAUBnCDRm (master) = $0000_H$**

- When a valid TAUBTTINm input edge is detected, the value $0000_H$ is written to TAUBnCNTm (master). The counter is set to $0000_H$ for one count and returns to $FFFF_H$.
  Thus, the slave channel starts to count down one count clock later to TAUBTTINm (master).

### (2) TAUBnCDRm (slave) = $0000_H$

The following settings apply to this diagram.

- Start trigger detection disabled during counting (TAUBnCMORm.TAUBnMD0 = 0)
- Falling edge detection (TAUBnCMURm.TAUBnTIS[1:0] = $00_B$)

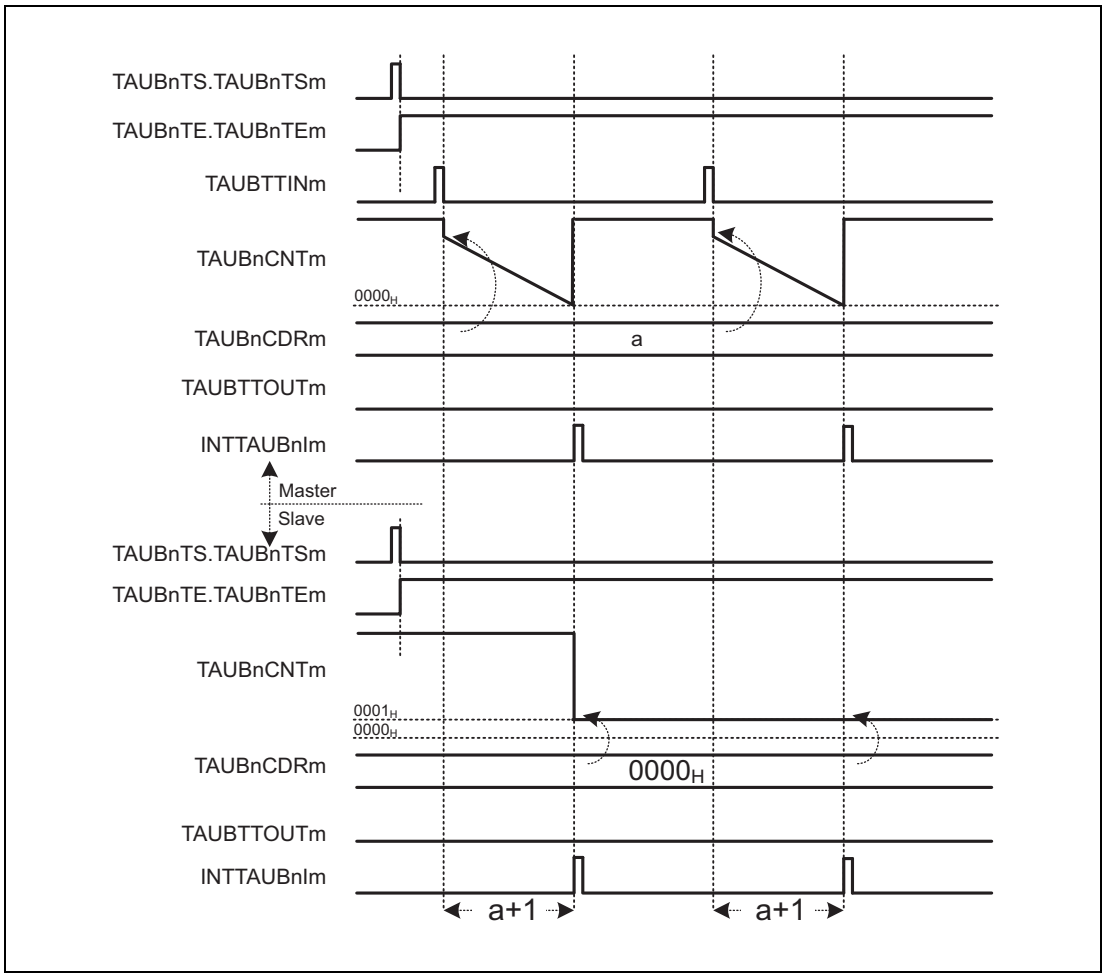



**Figure 24.87 TAUBnCDRm (slave) = $0000_H$**

- TAUBTTOUTm remains at inactive state, because the pulse width is zero.

### (3) TAUBnCMORm.TAUBnMD0 = 0

The following settings apply to this diagram.

- Start trigger detection disabled during counting (TAUBnCMORm.TAUBnMD0 = 0)
- Falling edge detection (TAUBnCMURm.TAUBnTIS[1:0] = $00_B$)

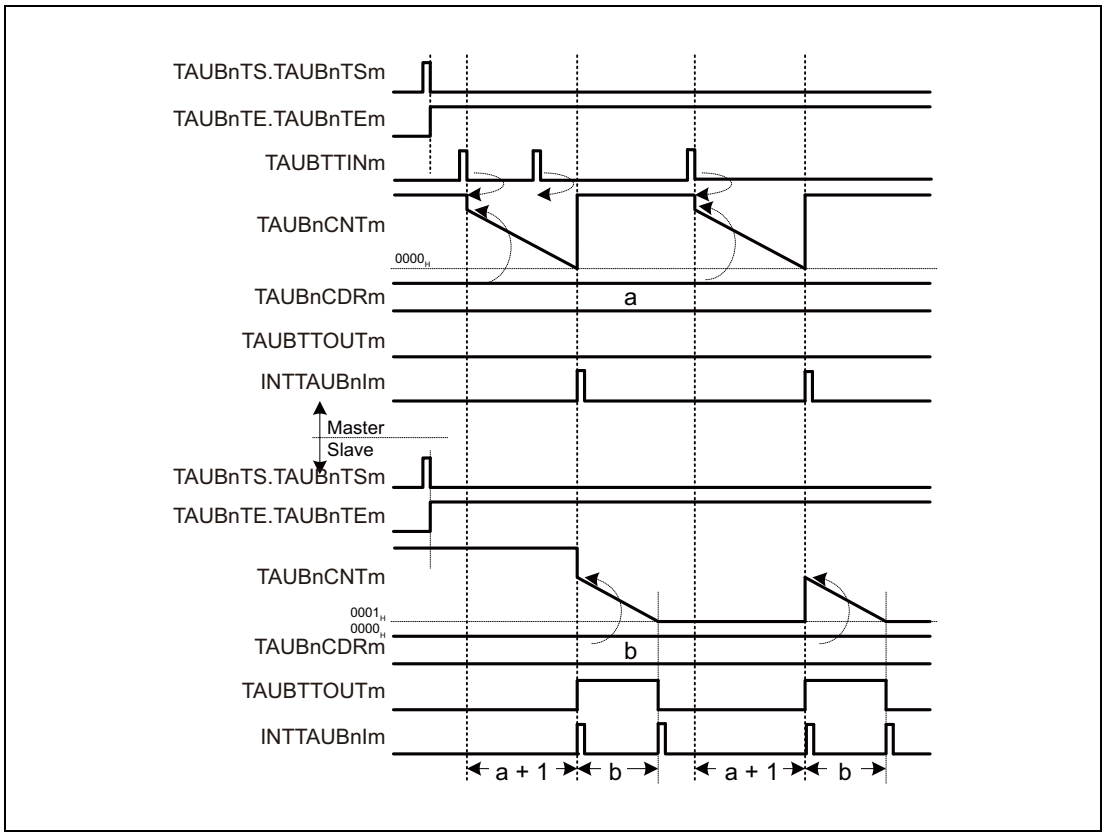



**Figure 24.88 TAUBnCMORm.TAUBnMD0 = 0**

- Even when an effective edge is input to TAUBTTINm while the counter of the master channel counts down, the counter continues counting down.

### (4) TAUBnCMORm.TAUBnMD0 = 1

The following settings apply to this diagram.

- Start trigger detection enabled during counting (TAUBnCMORm.TAUBnMD0 = 1)
- Falling edge detection (TAUBnCMURm.TAUBnTIS[1:0] = $00_B$)

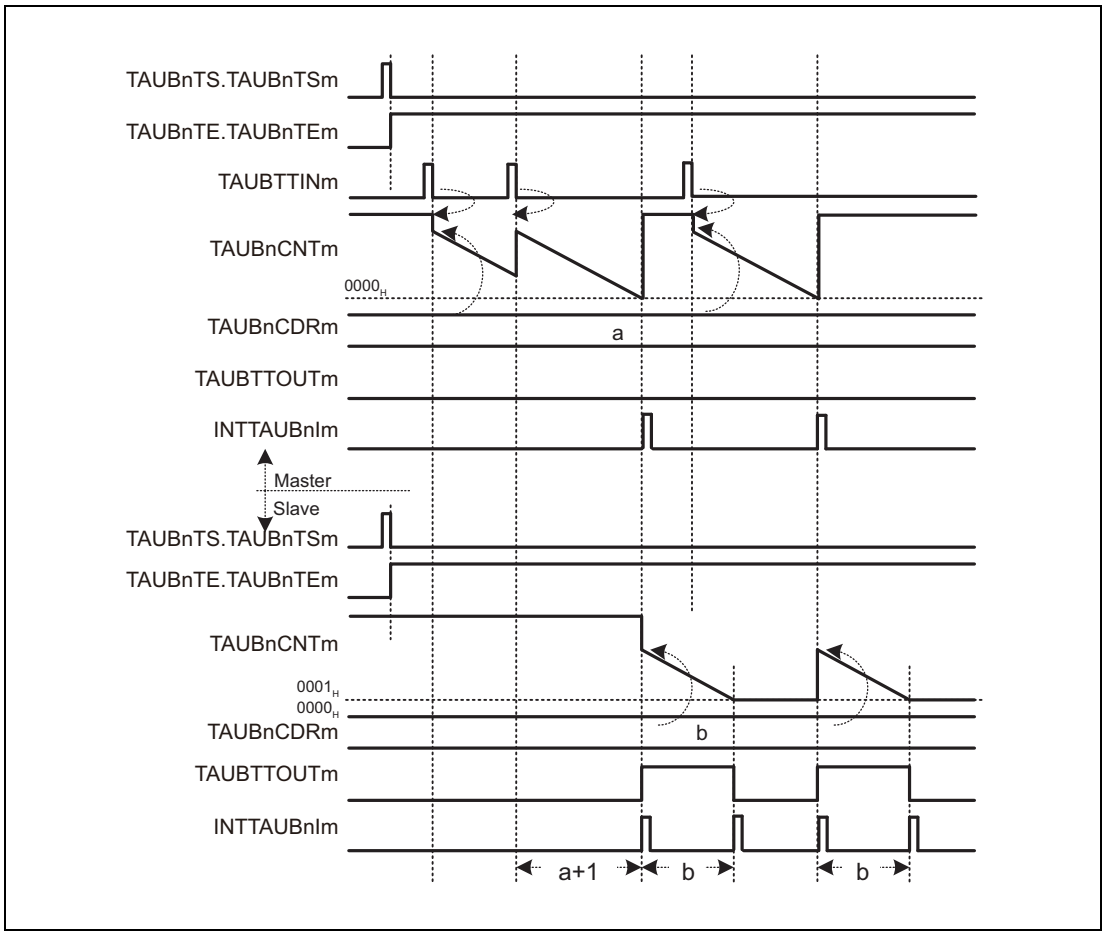



**Figure 24.89 TAUBnCMORm.TAUBnMD0 = 1**

- If a valid TAUBTTINm input edge is detected while the counter of the master channel counts down, TAUBnCNTm reloads the value of TAUBnCDRm. The counter restarts to count down.

This means the delay for INTTAUBnIm generation interval is extended by the value of TAUBnCNTm at the time when a valid TAUBTTINm input edge is detected.

### (5) Operation stop and restart

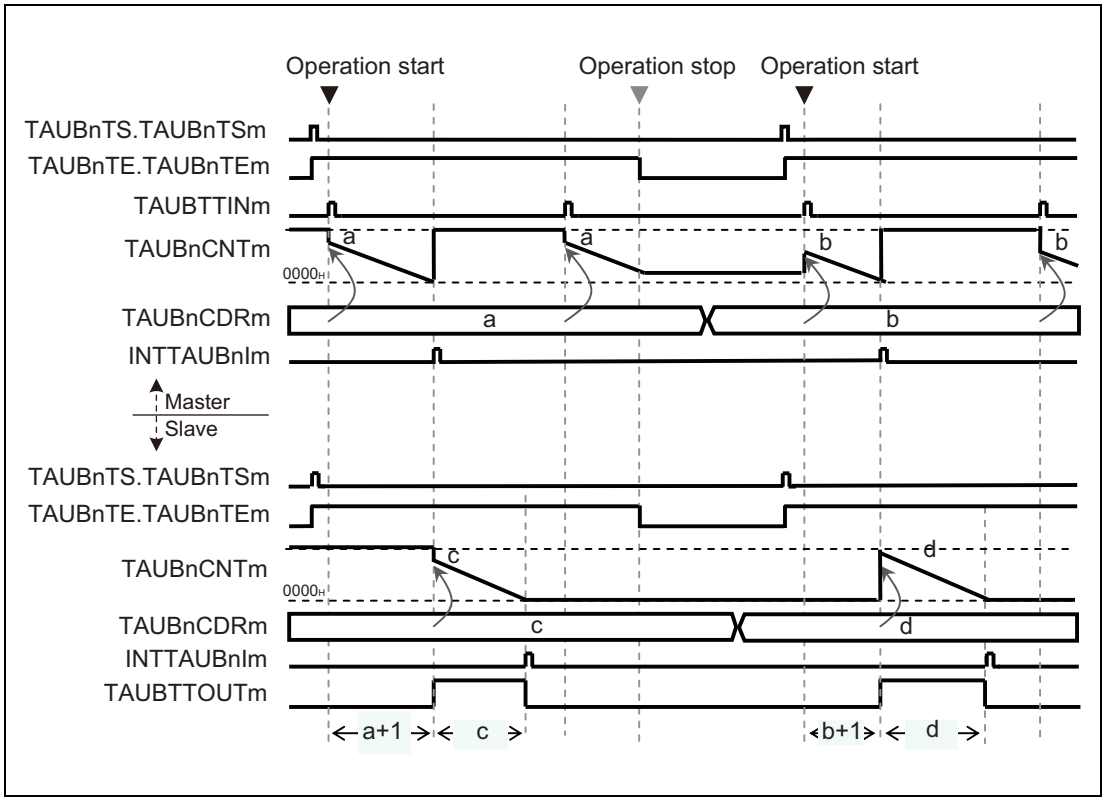



**Figure 24.90 Stopping and Restarting the Operation**

Setting TTm of the master and slave channels to 1 clears TAUBnTE.TAUBnTEm to 0, thereby stopping the count operation. If this happens, TAUBnCNTm and TAUBTTOUTm stop operation with the values retained.

Setting TAUBnTS.TAUBnTSm of the master and slave channels to 1 concurrently sets TAUBnTE.TAUBnTEm to 1.

When the start trigger is detected while the TAUBnTE.TAUBnTEm is set to 1, the TAUBnCDRm value is transferred to TAUBnCNTm and the operation restarts.

### (6) Restarting the master channel while the slave channel is counting

The following settings apply to this diagram.

- Start trigger detection disabled during counting (TAUBnCMORm.TAUBnMD0 = 0)
- Falling edge detection (TAUBnCMURm.TAUBnTIS[1:0] = $00_B$)

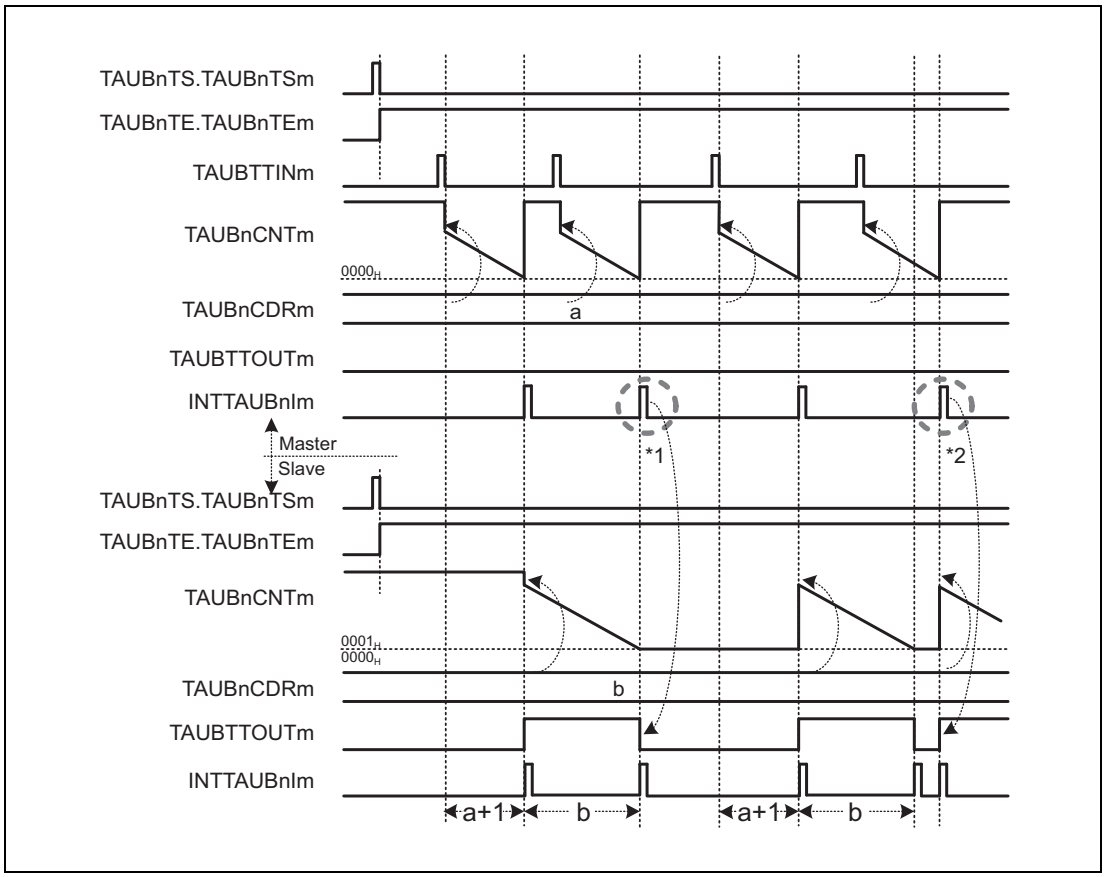



**Figure 24.91 Input Interval of TAUBTTINm ≤ Delay Time + Pulse Width + 1**

- If the master channel generates an interrupt before the counter of the slave channel has reached $0001_H$ or exactly when $0001_H$ is reached (*1), the interrupt (master) is ignored.
- If an interrupt of the master channel occurs when the counter of the slave channel awaits the next trigger, the value of TAUBnCDRm (slave) is reloaded. An interrupt is generated and TAUBTTOUTm toggles. If TAUBnCNTm (master) has started to count down while the TAUBnCNTm (slave) is still counting (*2), TAUBTTOUTm is not output with the expected delay time.
- To generate the correct one-shot pulse, the start trigger for the master channel must be detected while the master and slave channels are waiting for the start trigger, and not while they are counting.

## 24.14.3 Delay Pulse Output Function

### 24.14.3.1 Overview

#### Summary

This function outputs two signals. The reference signal has a defined pulse width and pulse cycle specified using the master channel and slave channel 1.
Slave channels 2 and 3 output the reference signal with a specified delay. The delay signal is identical to the reference signal, but delayed by amount specified in slave channel 2.

The signal values are specified in the following way:

- The pulse cycle is specified using the master channel.
- The duty cycle of the reference signal is specified using slave channel 1.
  The duty cycle of the delay signal is specified using slave channel 3.
- The delay is specified in slave channel 2.

#### Prerequisites

- Four channels
- The operation mode of the master channel must be set to interval timer mode, see **Table 24.115, Contents of the TAUBnCMORm Register for the Master Channel of the Delay Pulse Output Function**.
- The operation mode of slave channels 1 and 2 must be set to one-count mode, see **Table 24.118, Contents of the TAUBnCMORm Register for the Slave Channel 1 of the Delay Pulse Output Function** and **Table 24.122, Contents of the TAUBnCMORm Register for the Slave Channel 2 of the Delay Pulse Output Function**.
- The operation mode of slave channel 3 must be set to pulse one-count mode, see **Table 24.125, Contents of the TAUBnCMORm Register for the Slave Channel 3 of the Delay Pulse Output Function**
- TAUBTTOUTm is not used for the master channel and slave channel 2
- The channel output mode of slave channel 1 must be set to synchronous channel output mode 1.
- The channel output mode of slave channel 3 must be set to independent channel output mode 1.

#### Description

The counters of the channel group are enabled by setting the channel trigger bit (TAUBnTS.TAUBnTSm) to 1. This in turn sets TAUBnTE.TAUBnTEm to 1, enabling count operation.

- Master channel:

  The current value of TAUBnCDRm is written to TAUBnCNTm and the counter starts to count down from this value. INTTAUBnIm is generated on the master channel.
  When the counter of the master channel reaches $0000_H$, pulse cycle time has elapsed and INTTAUBnIm is generated. The counter reloads the TAUBnCDRm value and counts down.

- Slave channels 1 and 2:

  When the slave channels 1 and 2 detect an interrupt from the master channel, they start to count down from the current value of TAUBnCDRm. The TAUBTTOUTm signal (slave 1) is set.

  – Slave channel 1:
  When the counter of slave channel 1 reaches $0000_H$ (duty time has elapsed) INTTAUBnIm is generated and the TAUBTTOUTm signal is reset. The counter returns to $FFFF_H$ and awaits the next INTTAUBnIm of the master channel.

  – Slave channel 2:
  When the counter of slave channel 2 reaches $0000_H$, delay time has elapsed and INTTAUBnIm is generated. The counter returns to $FFFF_H$ and awaits the next INTTAUBnIm of the master channel.
  INTTAUBnIm (slave channel 2) triggers the counter of slave channel 3

- Slave channel 3:
  When slave channel 3 detects an interrupt from slave channel 2, it starts to count down from the current value of TAUBnCDRm. INTTAUBnIm is generated and the TAUBTTOUTm signal (slave 3) is set.
  When the counter of slave channel 3 reaches $0001_H$, INTTAUBnIm is generated and the TAUBTTOUTm signal is reset.
  The output from slave channel 3 is the delayed PWM pulse.

The counter can be stopped by setting TAUBnTT.TAUBnTTm to 1 for the master and slave channels, which in turn sets TAUBnTE.TAUBnTEm to 0. TAUBnCNTm and TAUBTTOUTm of master and slave channels stop but retain their values. The counters can be restarted by setting TAUBnTS.TAUBnTSm to 1.

**Conditions**

Simultaneous rewrite can be used with this function. Please see **Section 24.6, Simultaneous Rewrite**.

### 24.14.3.2 Equations

Pulse cycle = (TAUBnCDRm (master) + 1) × count clock cycle

Duty width 1 = (TAUBnCDRm (slave 1)) × count clock cycle

Delay = (TAUBnCDRm (slave 2) + 1) × count clock cycle

Duty width 2 = (TAUBnCDRm (slave 3)) × count clock cycle

Where the setting of the delay is within the following range:

$0000_H \leq$ TAUBnCDRm (slave 2) < TAUBnCDRm (master)

**NOTES**

1. The output waveform of TAUBTTOUTm (slave 3) is the output waveform of TAUBTTOUTm (slave 1) delayed for the delay generated by slave 2. It cannot be delayed for more than the pulse cycle.
2. When INTTAUBnIm of slave 2 occurs while slave 3 is counting, slave 3 restarts the operation. Therefore, the output waveform of TAUBTTOUTm (slave 3) retains the active level. (In this case, TAUBTTOUTm (Slave-CH-3) cannot output the waveform of the delayed basic pulse of TAUBTTOUTm (Slave-CH-1).)

### 24.14.3.3 Block Diagram and General Timing Diagram

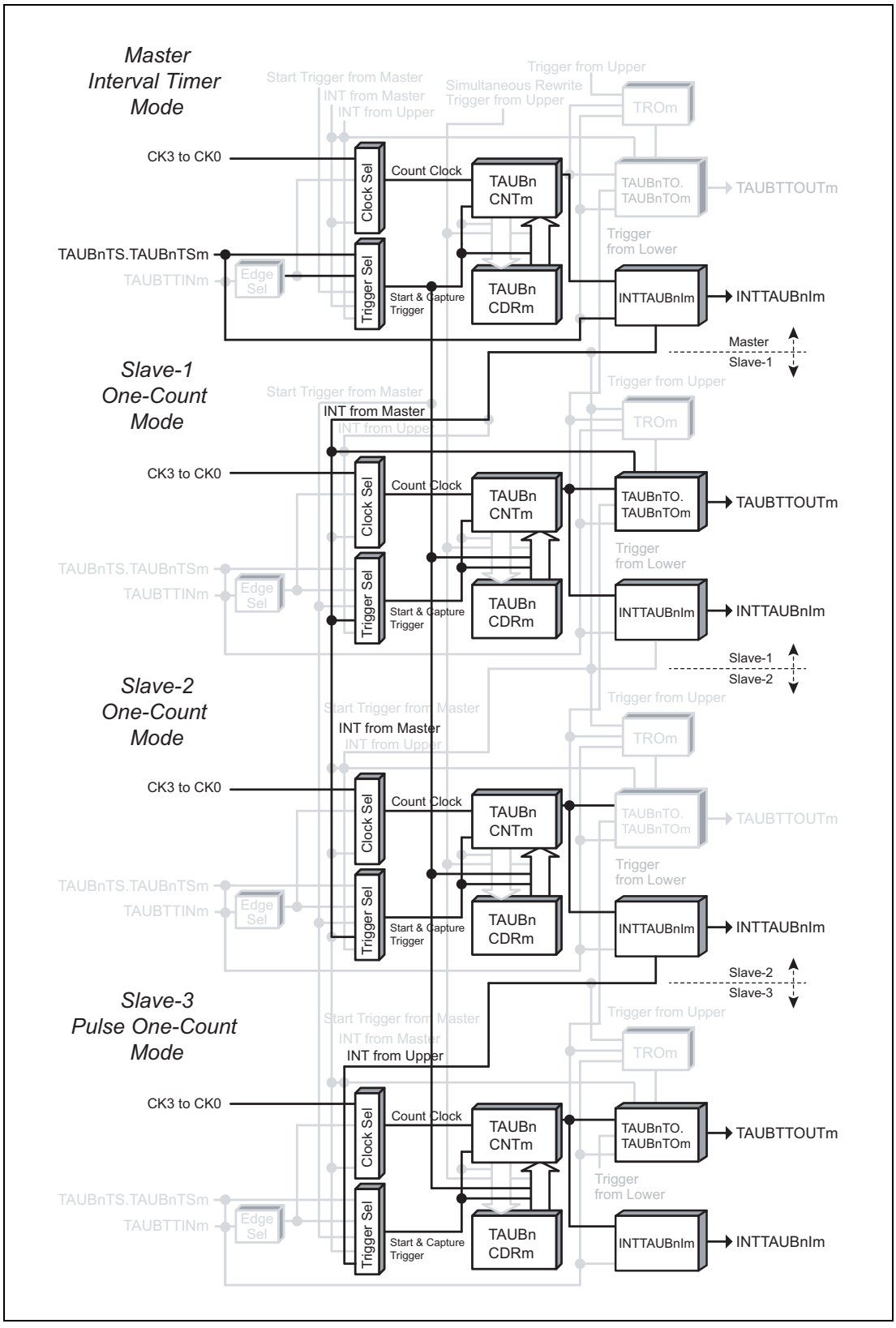



Figure 24.92 Block Diagram for Delay Pulse Output Function

The following settings apply to the general timing diagram.

- Slave channel 1: Positive logic (TAUBnTOL.TAUBnTOLm = 0)
- Slave channel 3: Positive logic (TAUBnTOL.TAUBnTOLm = 0)

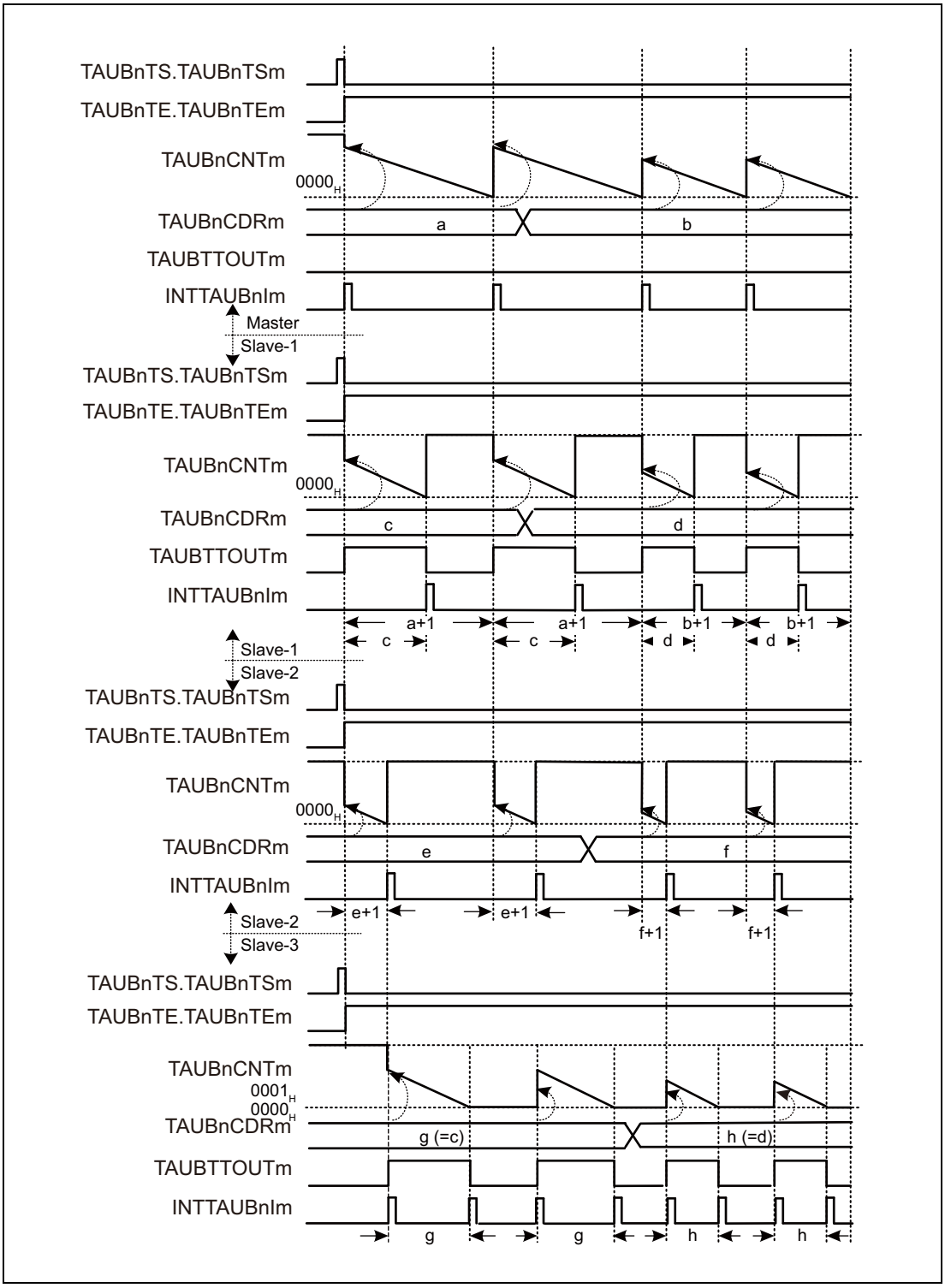



**Figure 24.93 General Timing Diagram for Delay Pulse Output Function**

### 24.14.3.4 Register Settings for the Master Channel

#### (1) TAUBnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.115 Contents of the TAUBnCMORm Register for the Master Channel of the Delay Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUBnCKS[1:0] bits of the master and slave channels must be identical. |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $1_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $000_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0000_B$. |
| 0 | TAUBnMD0 | Write $1_B$. |

#### (2) TAUBnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.116 Contents of the TAUBnCMURm Register for the Master Channel of the Delay Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Not used, so set to 00. |

### (3) Channel output mode for the master channel

Set TAUBnTOE.TAUBnTOEm to 0 because the channel output mode is not used by the master channel of this function.

### (4) Simultaneous rewrite for the master channel

The simultaneous rewrite settings of the master and slave channels must be identical.

**Table 24.117 Simultaneous Rewrite Settings for the Master Channel of the Delay Pulse Output Function**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 1: Enables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: The master channel is the control channel for simultaneous rewrite |
| TAUBnRDM.TAUBnRDMm | 0: The simultaneous rewrite trigger signal is generated when the master channel starts counting |
| TAUBnRDC.TAUBnRDCm | 0: Does not use the channel to generate the simultaneous rewrite trigger. |

### 24.14.3.5 Register Settings for Slave Channel 1

#### (1) TAUBnCMORm for slave channel 1

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.118 Contents of the TAUBnCMORm Register for the Slave Channel 1 of the Delay Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUBnCKS[1:0] bits of the master and slave channels must be identical. |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $100_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0100_B$. |
| 0 | TAUBnMD0 | Write $1_B$. |

#### (2) TAUBnCMURm for slave channel 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.119 Contents of the TAUBnCMURm Register for the Slave Channel 1 of the Delay Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Not used, so set to 00. |

### (3) Channel output mode for slave channel 1

Table 24.120 Control Bit Settings for Slave Channel 1 of the Synchronous Channel Output Mode 1

| Bit Name | Setting |
|---|---|
| TAUBnTOE.TAUBnTOEm | Write $1_B$. |
| TAUBnTOM.TAUBnTOMm | Write $1_B$. |
| TAUBnTOC.TAUBnTOCm | Write $0_B$. |
| TAUBnTOL.TAUBnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUBnTDE.TAUBnTDEm | Write $0_B$. |
| TAUBnTDL.TAUBnTDLm | Write $0_B$. |

### (4) Simultaneous rewrite for slave channel 1

The simultaneous rewrite settings of the master and slave channels must be identical.

Table 24.121 Simultaneous Rewrite Settings for Slave Channel 1 of the Delay Pulse Output Function

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 1: Enables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: The master channel is the control channel for simultaneous rewrite |
| TAUBnRDM.TAUBnRDMm | 0: The simultaneous rewrite trigger signal is generated when the master channel starts counting |
| TAUBnRDC.TAUBnRDCm | 0: Does not use the channel to generate the simultaneous rewrite trigger. |

### 24.14.3.6 Register Settings For Slave Channel 2

#### (1) TAUBnCMORm for slave channel 2

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.122 Contents of the TAUBnCMORm Register for the Slave Channel 2 of the Delay Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUBnCKS[1:0] bits of the master and slave channels must be identical. |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $100_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0100_B$. |
| 0 | TAUBnMD0 | Write $1_B$. |

#### (2) TAUBnCMURm for slave channel 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.123 Contents of the TAUBnCMURm Register for the Slave Channel 2 of the Delay Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Not used, so set to 00. |

### (3) Channel output mode for slave channel 2

Set TAUBnTOE.TAUBnTOEm to 0 because the channel output mode is not used by this function.

### (4) Simultaneous rewrite for slave channel 2

The simultaneous rewrite settings of the master and slave channels must be identical.

**Table 24.124 Simultaneous Rewrite Settings for Slave Channel 2 of the Delay Pulse Output Function**

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 1: Enables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: The master channel is the control channel for simultaneous rewrite |
| TAUBnRDM.TAUBnRDMm | 0: The simultaneous rewrite trigger signal is generated when the master channel starts counting |
| TAUBnRDC.TAUBnRDCm | 0: Does not use the channel to generate the simultaneous rewrite trigger. |

### 24.14.3.7 Register Settings for Slave Channel 3

#### (1) TAUBnCMORm for slave channel 3

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.125 Contents of the TAUBnCMORm Register for the Slave Channel 3 of the Delay Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUBnCKS[1:0] bits of the master and slave channels must be identical. |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $101_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $1010_B$. |
| 0 | TAUBnMD0 | Write $1_B$. |

#### (2) TAUBnCMURm for slave channel 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.126 Contents of the TAUBnCMURm Register for the Slave Channel 3 of the Delay Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Not used, so set to 00. |

#### (3) Channel output mode for slave channel 3

Table 24.127 Control Bit Settings for Independent Channel Output Mode 2

| Bit Name | Setting |
|---|---|
| TAUBnTOE.TAUBnTOEm | Write $1_B$. |
| TAUBnTOM.TAUBnTOMm | Write $0_B$. |
| TAUBnTOC.TAUBnTOCm | Write $1_B$. |
| TAUBnTOL.TAUBnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUBnTDE.TAUBnTDEm | Write $0_B$. |
| TAUBnTDL.TAUBnTDLm | Write $0_B$. |

#### (4) Simultaneous rewrite for slave channel 3

The simultaneous rewrite settings of the master and slave channels must be identical.

Table 24.128 Simultaneous Rewrite Settings for Slave Channel 3 of the Delay Pulse Output Function

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 1: Enables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: The master channel is the control channel for simultaneous rewrite |
| TAUBnRDM.TAUBnRDMm | 0: The simultaneous rewrite trigger signal is generated when the master channel starts counting |
| TAUBnRDC.TAUBnRDCm | 0: Does not use the channel to generate the simultaneous rewrite trigger. |

### 24.14.3.8 Operating Procedure for Delay Pulse Output Function

Table 24.129 Operating Procedure for Delay Pulse Output Function (1/2)

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Master channel: set the TAUBnCMORm and TAUBnCMURm registers and the channel output mode as described in **Section 24.14.3.4, Register Settings for the Master Channel**.<br><br>Slave channel 1: set the TAUBnCMORm and TAUBnCMURm registers and the channel output mode as described in **Section 24.14.3.5, Register Settings for Slave Channel 1**.<br><br>Slave channel 2: set the TAUBnCMORm and TAUBnCMURm registers and the channel output mode as described in **Section 24.14.3.6, Register Settings For Slave Channel 2**.<br><br>Slave channel 3: set the TAUBnCMORm and TAUBnCMURm registers and the channel output mode as described in **Section 24.14.3.7, Register Settings for Slave Channel 3**.<br><br>Set the values of the TAUBnCDRm registers of all channels | Channel operation is stopped. |

**Table 24.129 Operating Procedure for Delay Pulse Output Function (2/2)**

| | Operation | Status of TAUBn |
|---|---|---|
| Start operation | Set TAUBnTS.TAUBnTSm of the master and slave channels to 1 simultaneously.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm (master and slave channels) is set to 1 and the counters of the master channel and slave channels 1 and 2 start. INTTAUBnIm is generated on the master channel and TAUBTTOUTm (slave 1) is set. |
| During operation | TAUBnCDRm can be changed at any time. TAUBnCNTm and TAUBnRSF.TAUBnRSFm can be read at any time.<br><br>TAUBnRDT.TAUBnRDTm can be changed during operation. | TAUBnCNTm of the master channel and slave channels 1 and 2 load TAUBnCDRm and count down.<br><br>When the counter of the master channel reaches $0000_H$:<br>• INTTAUBnIm (master) is generated<br>• TAUBnCNTm (master) reloads the TAUBnCDRm value and continues count operation<br>• TAUBnCNTm (slave 1 and slave 2) reload the TAUBnCDRm value and start counting down<br>• TAUBTTOUTm (slave 1) is set<br><br>When TAUBnCNTm (slave 1) reaches $0000_H$:<br>• INTTAUBnIm (slave 1) is generated<br>• TAUBTTOUTm (slave 1) is reset<br><br>When TAUBnCNTm (slave 2) reaches $0000_H$:<br>• INTTAUBnIm (slave 2) is generated<br>• INTTAUBnIm (slave 3) is generated<br>• TAUBTTOUTm (slave 3) is set<br>• TAUBnCNTm (slave 3) reloads the TAUBnCDRm value and starts counting down<br><br>When TAUBnCNTm (slave 3) reaches $0001_H$:<br>• INTTAUBnIm (slave 3) is generated<br>• TAUBTTOUTm (slave 3) is reset |
| Stop operation | Set TAUBnTT.TAUBnTTm of the master and slave channels to 1 simultaneously.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm and TAUBTTOUTm stop and retain their current values. |

Restart operation

### 24.14.3.9 Specific Timing Diagrams

#### (1) Duty cycle (slave 3) = 100%

The following values apply to the figure below.

- TAUBnCDRm (master) = $000A_H$
- TAUBnCDRm (slave 1) = $000B_H$
- TAUBnCDRm (slave 2) = $0000_H$
- TAUBnCDRm (slave 3) = $000B_H$

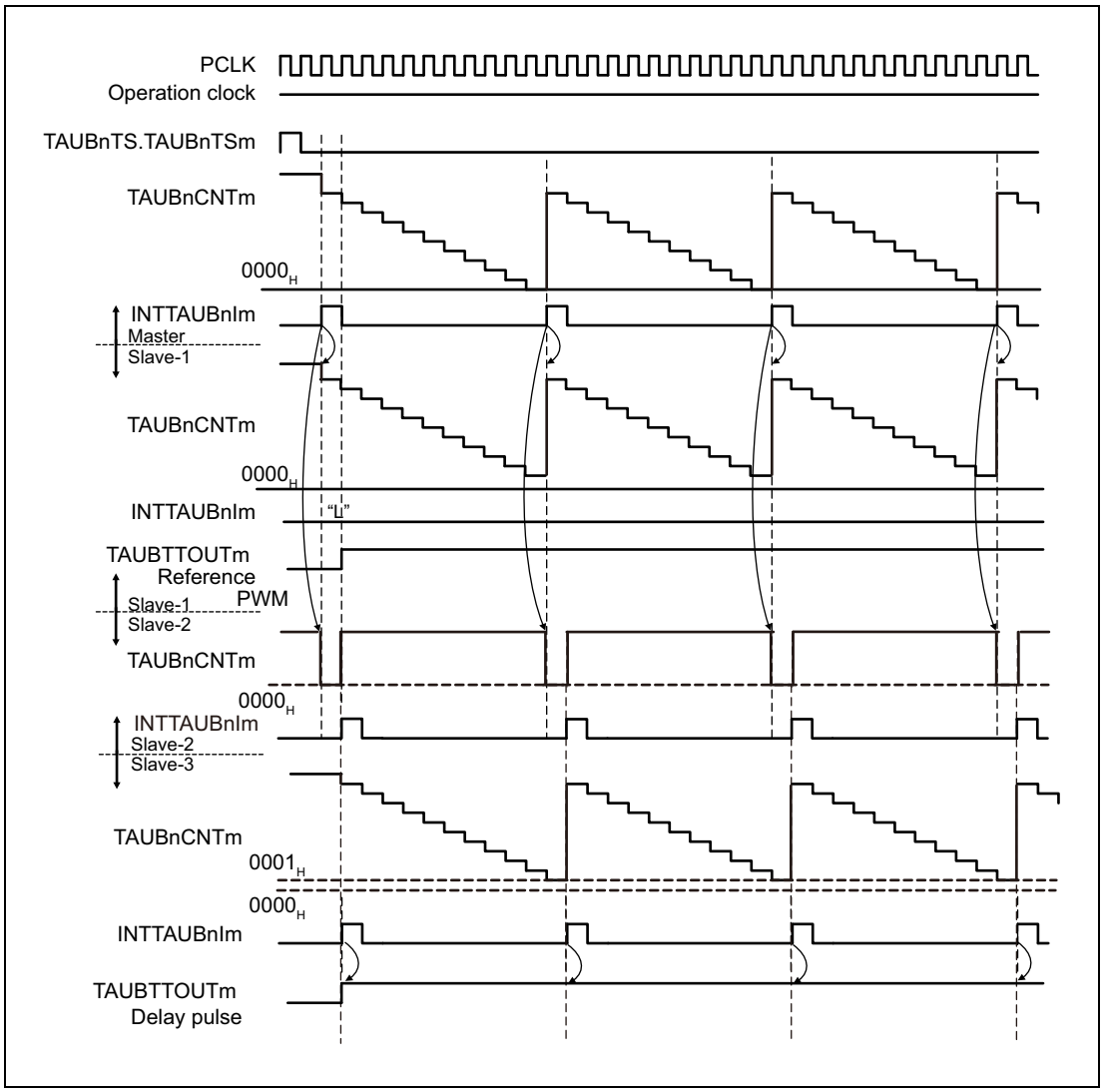



**Figure 24.94 Duty Cycle (slave 3) = 100%**

- If the value of TAUBnCDRm (slave 1 and 3) is higher than the value of TAUBnCDRm (master), the counter of slave channel 1 cannot reach 0000H and cannot generate interrupt request signals. TAUBTTOUTm of channels 1 and 3 remain in the active state.

### (2) TAUBTTOUTm (slave 1) = TAUBTTOUTm (slave 3)

The following values apply to the figure below.

- TAUBnCDRm (master) = $000A_H$
- TAUBnCDRm (slave 1) = $0005_H$
- TAUBnCDRm (slave 2) = $0000_H$
- TAUBnCDRm (slave 3) = $0005_H$

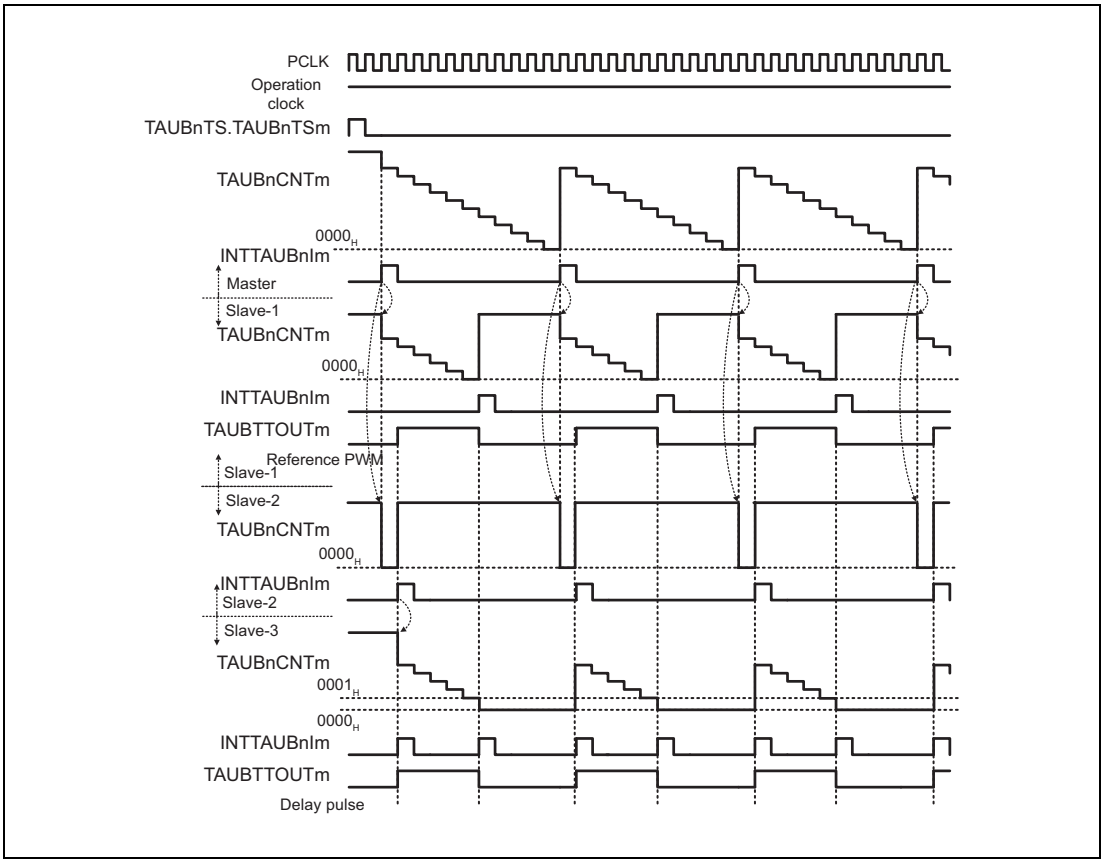



**Figure 24.95 TAUBTTOUTm (slave 1) = TAUBTTOUTm (slave 3)**

- If TAUBnCDRm (slave 2) = $0000_H$, the counter of slave channel 3 starts counting one count clock later than the counter of slave channel 1. The reference pulse and the delay pulse are output with a delay of one clock count.

## 24.14.4 AD Conversion Trigger Output Function Type 1

### 24.14.4.1 Overview

#### Summary

This function is identical to **Section 24.14.1, PWM Output Function** except that TAUBTTOUTm is not output.

This is achieved by setting the channel output mode of the slave to independent channel output mode controlled by software.

### 24.14.4.2 Block Diagram and General Timing Diagram

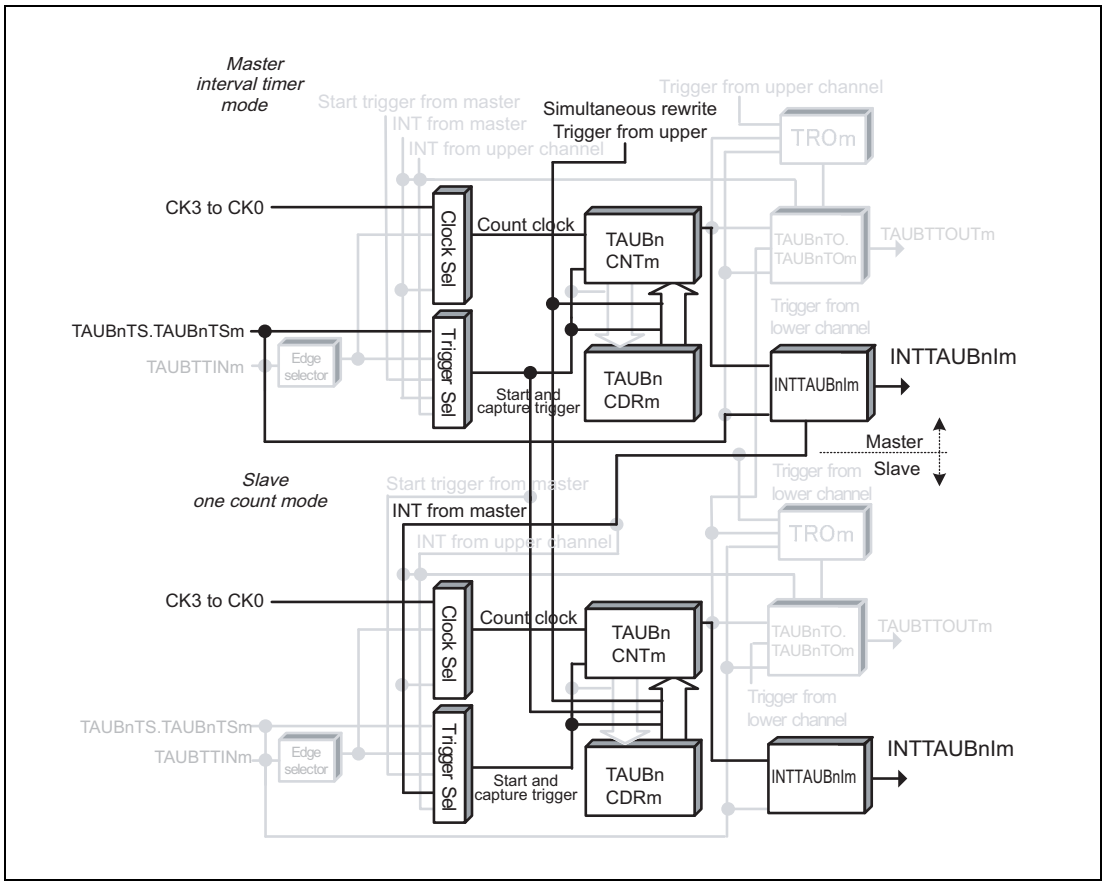



**Figure 24.96 Block Diagram for AD Conversion Trigger Output Function Type 1**

The following settings apply to the general timing diagram.

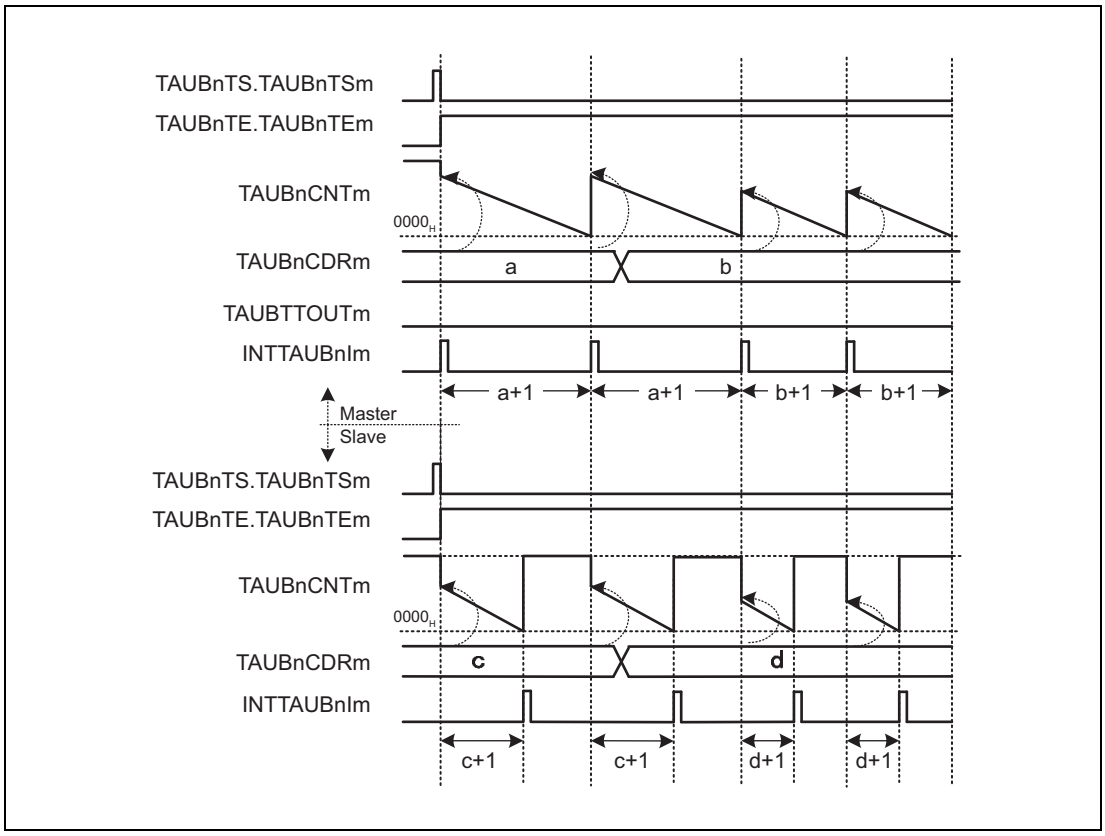



**Figure 24.97 General Timing Diagram for AD Conversion Trigger Output Function Type 1**

## 24.14.5 Triangle PWM Output Function

### 24.14.5.1 Overview

#### Summary

This function generates multiple triangle PWM outputs by using a master and one or more slave channels. It enables the pulse cycle (frequency) and the duty cycle of TAUBTTOUTm to be set using the master and slave channel(s) respectively.

The master channel generates a carrier cycle from two pulse cycles. The first cycle of the master channel controls the down status and the second cycle controls the up status of the slaves counter.

#### Prerequisites

- Two channels
- The operation mode of the master channel must be set to interval timer mode, see **Table 24.130, Contents of the TAUBnCMORm Register for the Master Channel of the Triangle PWM Output Function**.
- The operation mode of the slave channel(s) must be set to up down count mode, see **Table 24.134, Contents of the TAUBnCMORm Register for the Slave Channel of the Triangle PWM Output Function**.
- The channel output mode of the master channel must be set to independent channel output mode 1.
- The channel output mode of the slave channel(s) must be set to synchronous channel output mode 2.
- The following settings establish TAUBTTOUTm at high level for the down status of the carrier cycle.
  - If the TAUBnCMORm.TAUBnMD0 (master) bit is set to 0,TAUBnTO.TAUBnTOm must be set to 1 while TAUBnTOE.TAUBnTOEm is 0. (recommended)
  - If the TAUBnCMORm.TAUBnMD0 (master) bit is set to 1, TAUBnTO.TAUBnTOm must be set to 0 while TAUBnTOE.TAUBnOEm is 0.

#### Functional description

The counters are enabled by setting the channel trigger bit (TAUBnTS.TAUBnTSm) to 1 for every channel. This in turn sets TAUBnTE.TAUBnTEm, enabling count operation. The current values of TAUBnCDRm (master and slave) are written to TAUBnCNTm (master and slave) and the counters start to count down from these values. If the master channel TAUBnCMORm.TAUBnMD0 bit is set to 1, an interrupt is generated and TAUBTTOUTm signal of the master toggles.

- Master channel:
  When the counter of the master channel reaches $0000_H$ (pulse cycle time has elapsed) INTTAUBnIm is generated and the TAUBTTOUTm signal toggles. TAUBnCNTm then reloads the TAUBnCDRm value and counts down.

- Slave channel:
  INTTAUBnIm generated on the master channel triggers the counter of the slave channel:
  - If the slave counter currently counts down, it changes count direction.
  - If the slave counter currently counts up, the value of TAUBnCDRm is reloaded and the counter counts down.

  When the counter of the slave channel reaches $0001_H$ while counting up or down, INTTAUBnIm is generated and the TAUBTTOUTm (slave) signal is set or reset.

  The counter continues to count down or up and awaits the next INTTAUBnIm of the master channel.

  TAUBTTOUTm can be switched between positive and negative phase setting TAUBnTOL.TAUBnTOLm during operation.

The counters can be stopped by setting TAUBnTT.TAUBnTTm to 1 for the master and slave channel(s), which in turn sets TAUBnTE.TAUBnTEm to 0. TAUBnCNTm and TAUBTTOUTm of master and slave channel(s) stop but retain their values.

**Conditions**

Simultaneous rewrite can be used with this function. Please see **Section 24.6, Simultaneous Rewrite**.

### 24.14.5.2 Equations

Pulse cycle = (TAUBnCDRm (master) + 1) × count clock cycle

$0000_H \leq$ TAUBnCDRm (master) $< FFFF_H$

Carrier cycle (down/up) = (TAUBnCDRm (master) + 1) × 2 × count clock cycle

Duty cycle =

[(TAUBnCDRm (master) + 1 - TAUBnCDRm (slave))/ (TAUBnCDRm (master) + 1)] × 100

- Duty cycle = 100%
  TAUBnCDRm (slave) = $0000_H$
- Duty cycle = 0%
  TAUBnCDRm (slave) ≥ TAUBnCDRm (master) + 1

### 24.14.5.3 Block Diagram and General Timing Diagram

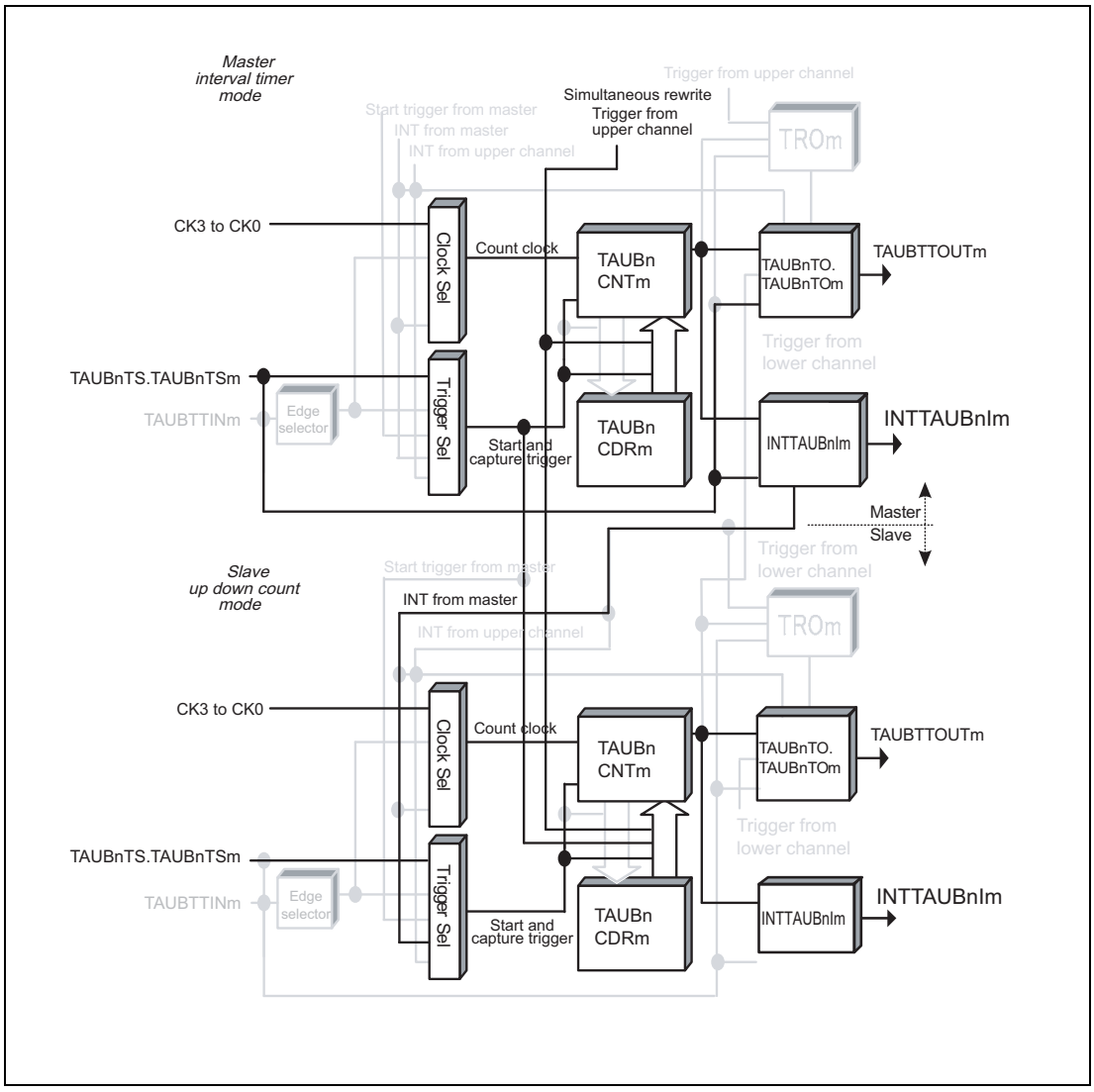



**Figure 24.98 Block Diagram for Triangle PWM Output Function**

The following settings apply to the general timing diagram.

- Master channel
  - INTTAUBnIm is generated at operation start (TAUBnCMORm.TAUBnMD0 = 1)

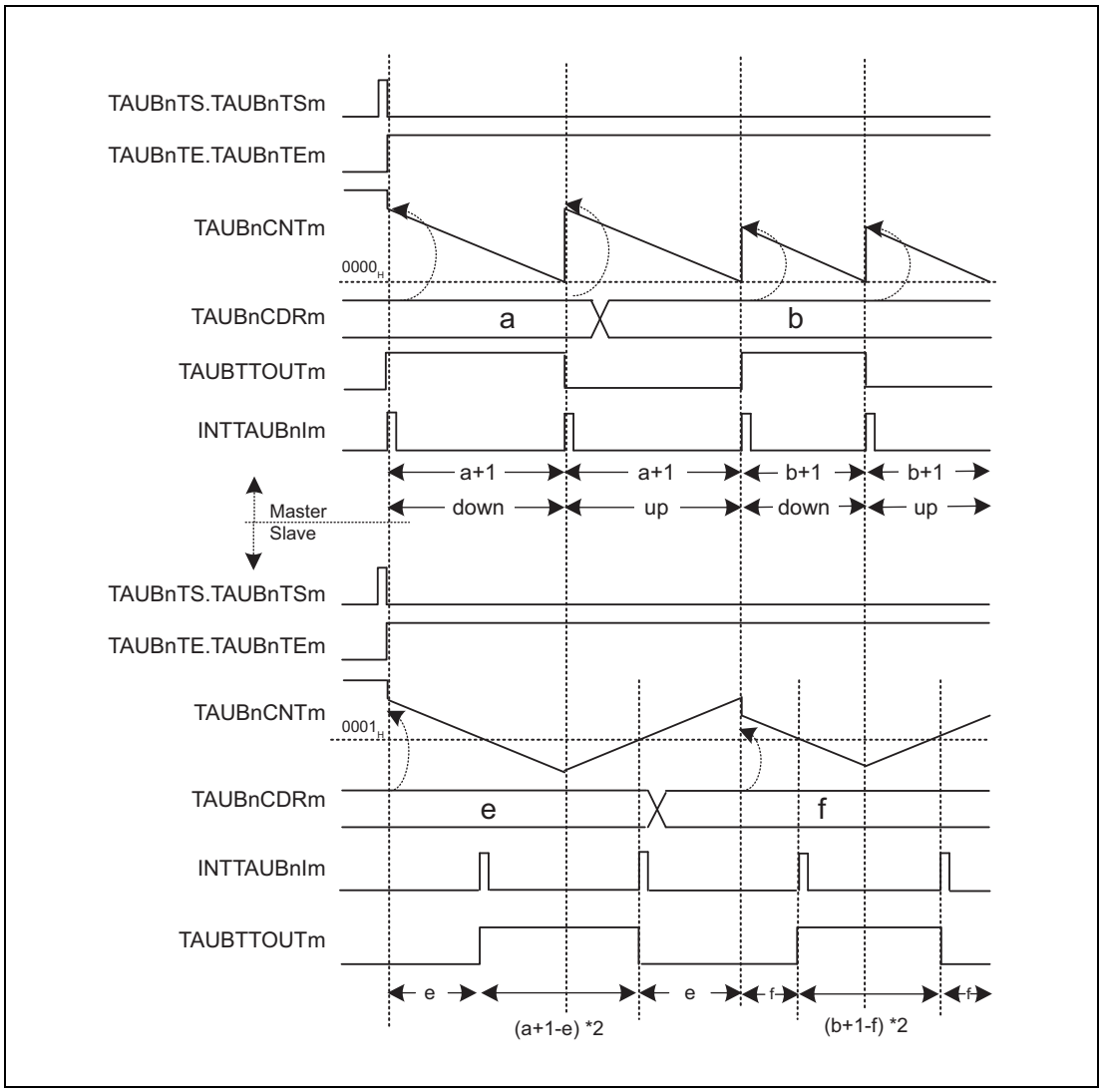



**Figure 24.99 General Timing Diagram for Triangle PWM Output Function**

### 24.14.5.4 Register Settings for the Master Channel

#### (1) TAUBnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.130 Contents of the TAUBnCMORm Register for the Master Channel of the Triangle PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUBnCKS[1:0] bits of the master and slave channel(s) must be identical. |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $1_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $000_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0000_B$. |
| 0 | TAUBnMD0 | 0: INTTAUBnIm not generated and TAUBTTOUTm does not toggle at operation start<br>1: Generates INTTAUBnIm and toggles TAUBTTOUTm at operation start |

#### (2) TAUBnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.131 Contents of the TAUBnCMURm Register for the Master Channel of the Triangle PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Not used, so set to 00. |

### (3) Channel output mode for the master channel

Table 24.132 Control Bit Settings for Independent Channel Output Mode 1

| Bit Name | Setting |
|---|---|
| TAUBnTOE.TAUBnTOEm | Write $1_B$. |
| TAUBnTOM.TAUBnTOMm | Write $0_B$. |
| TAUBnTOC.TAUBnTOCm | Write $0_B$. |
| TAUBnTOL.TAUBnTOLm | Write $0_B$. |
| TAUBnTDE.TAUBnTDEm | Write $0_B$. |
| TAUBnTDL.TAUBnTDLm | Write $0_B$. |

### (4) Simultaneous rewrite for the master channel

The simultaneous rewrite settings of the master and slave channels must be identical.

Table 24.133 Simultaneous Rewrite Settings for the Master Channel of the Triangle PWM Output Function

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 1: Enables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: Selects master channel for simultaneous rewrite triggers.<br>1: Selects upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUBnRDM.TAUBnRDMm | 1: The simultaneous rewrite trigger signal is generated when the master channel starts counting and the corresponding slave channel is at the peak of a triangular wave |
| TAUBnRDC.TAUBnRDCm | 0: Does not use the channel to generate the simultaneous rewrite trigger. |

**NOTE**

If TAUBnRDS.TAUBnRDSm = 1, the master channel requires an upper channel that generates the simultaneous rewrite trigger signal.

### 24.14.5.5 Register Settings for the Slave Channel(s)

#### (1) TAUBnCMORm for the slave channel(s)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.134 Contents of the TAUBnCMORm Register for the Slave Channel of the Triangle PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUBnCKS[1:0] bits of the master and slave channel(s) must be identical. |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $111_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $1001_B$. |
| 0 | TAUBnMD0 | Write $0_B$. |

#### (2) TAUBnCMURm for the slave channel(s)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.135 Contents of the TAUBnCMURm Register for the Slave Channel of the Triangle PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Not used, so set to 00. |

### (3) Channel output mode for the slave channel(s)

Table 24.136 Control Bit Settings for Synchronous Channel Output Mode 2

| Bit Name | Setting |
|---|---|
| TAUBnTOE.TAUBnTOEm | Write $1_B$. |
| TAUBnTOM.TAUBnTOMm | Write $1_B$. |
| TAUBnTOC.TAUBnTOCm | Write $1_B$. |
| TAUBnTOL.TAUBnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUBnTDE.TAUBnTDEm | Write $0_B$. |
| TAUBnTDL.TAUBnTDLm | Write $0_B$. |

### (4) Simultaneous rewrite for the slave channel(s)

The simultaneous rewrite settings of the master and slave channels must be identical.

Table 24.137 Simultaneous Rewrite Settings for the Slave Channel of the Triangle PWM Output Function

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 1: Enables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: Selects master channel for simultaneous rewrite triggers.<br>1: Selects upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUBnRDM.TAUBnRDMm | 1: The simultaneous rewrite trigger signal is generated when the master channel starts counting and the corresponding slave channel is at the peak of a triangular wave |
| TAUBnRDC.TAUBnRDCm | 0: Does not use the channel to generate the simultaneous rewrite trigger. |

### 24.14.5.6 Operating Procedure for Triangle PWM Output Function

Table 24.138 Operating Procedure for Triangle PWM Output Function

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Master channel: set the TAUBnCMORm and TAUBnCMURm registers and the channel output mode as described in **Section 24.14.5.4, Register Settings for the Master Channel**<br><br>Slave channel: set the TAUBnCMORm and TAUBnCMURm registers and the channel output mode as described in **Section 24.14.5.5, Register Settings for the Slave Channel(s)**<br><br>Set the values of the TAUBnCDRm registers of all channels | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm of the master and slave channels to 1 simultaneously.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm (master and slave channels) is set to 1 and the counters of the master and slave channels start.<br>INTTAUBnIm (master) is generated on the master channel when TAUBnCMORm.TAUBnMD0 set to 1. |
| During operation | TAUBnCDRm can be changed at any time.<br>TAUBnTOL.TAUBnTOLm can be changed.<br>TAUBnCNTm and TAUBnRSF.TAUBnRSFm can be read at any time.<br><br>TAUBnRDT.TAUBnRDTm can be changed during operation. | TAUBnCNTm of the master and slave channels loads TAUBnCDRm and counts down. When the counter of the master channel reaches $0000_H$:<br>• INTTAUBnIm (master) is generated<br>• TAUBTTOUTm (master) toggles<br>• TAUBnCNTm (master) reloads the TAUBnCDRm value and continues count operation.<br>• TAUBnCNTm (slave) reloads the TAUBnCDRm value or counts in the reverse direction.<br>When TAUBnCNTm of the slave = $0001_H$:<br>• INTTAUBnIm (slave) is generated<br>• TAUBTTOUTm (slave) is set (in count-down status) or reset (in count-up status) |
| Stop operation | Set TAUBnTT.TAUBnTTm of the master and slave channels to 1 simultaneously.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm and TAUBTTOUTm stop and retain their current values. |

Restart operation

### 24.14.5.7 Specific Timing Diagrams

#### (1) Duty cycle = 0%

The following settings apply to the general timing diagram.

- Master channel:
  - INTTAUBnIm is generated at operation start (TAUBnCMORm.TAUBnMD0 = 1)
  - TAUBnCDRm = a = $5_H$
- Slave channel:
  - TAUBnCDRm = $6_H$

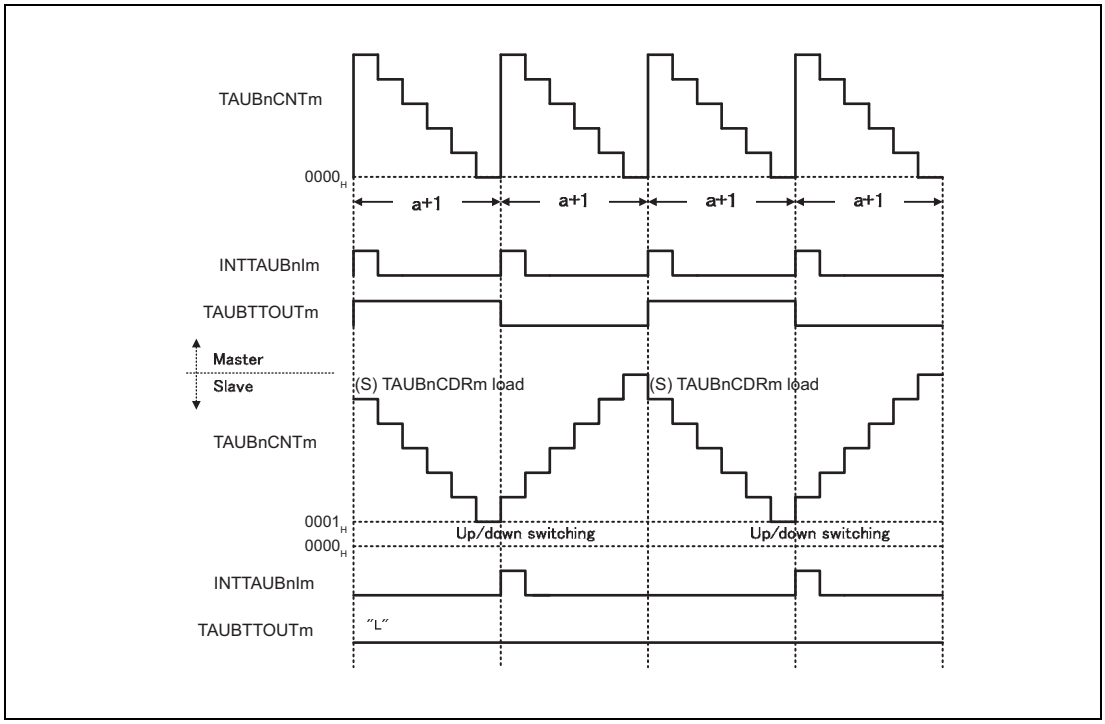



**Figure 24.100 TAUBnCDRm (slave) ≥ TAUBnCDRm (master) + 1**

- If TAUBnCDRm (slave) ≥ TAUBnCDRm (master) +1, INTTAUBnIm of slave channel is not generated during counting down. The set signal is never detected, so TAUBTTOUTm remains at low state.

### (2) Duty cycle = 100%

The following settings apply to the general timing diagram.

- Master channel:
  - INTTAUBnIm is generated at operation start (TAUBnCMORm.TAUBnMD0 = 1)
  - TAUBnCDRm = a = $5_H$
- Slave channel:
  - TAUBnCDRm = $0_B$

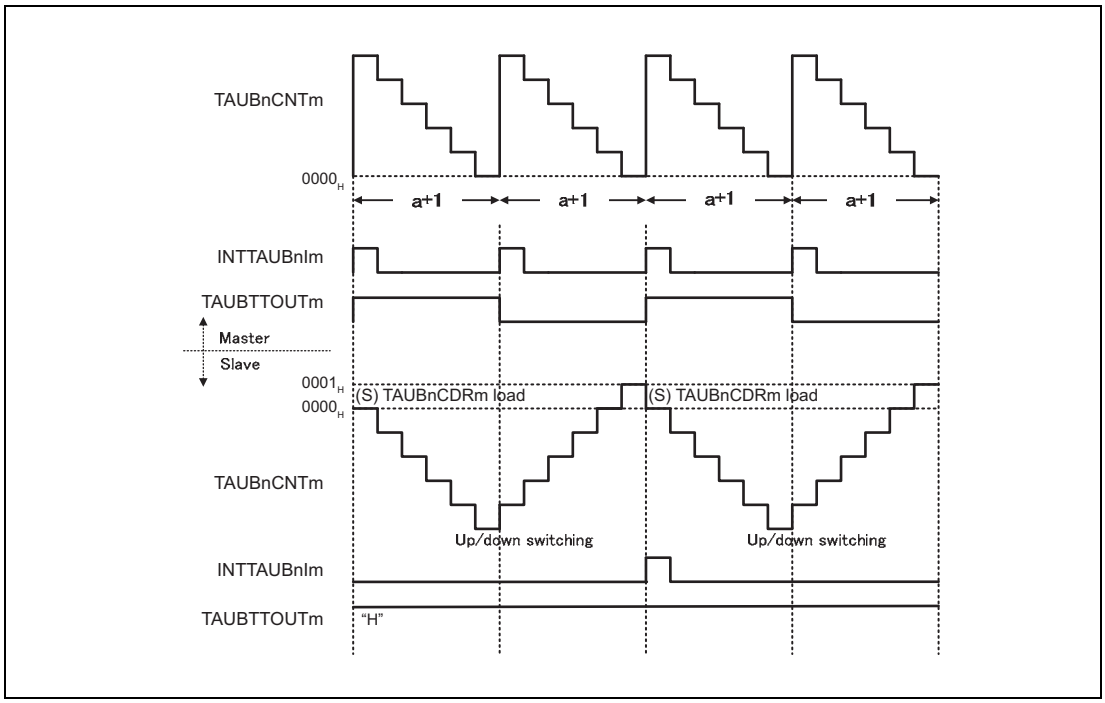



**Figure 24.101 TAUBnCDRm (slave) = $0000_H$**

- If TAUBnCDRm (slave) = $0000_H$, INTTAUBnIm of slave channel is not generated during counting up. The reset signal is never detected, so TAUBTTOUTm remains at high state.

## 24.14.6 Triangle PWM Output Function with Dead Time

### 24.14.6.1 Overview

#### Summary

This function generates multiple triangle PWM outputs with a defined dead time by using a master and two or more slave channels. The resulting PWM signals with the dead time are output via TAUBTTOUTm of the slave channels 2 and 3. It enables the pulse cycle (frequency) and the duty cycle of TAUBTTOUTm to be set using the master and slave channel(s) respectively.

The master generates a carrier cycle. The first pulse controls the down status and the second pulse controls the up status of the slaves counter.

An interrupt on slave 2 causes TAUBTTOUTm of the slave channels to be set or reset.
Depending on the settings of TAUBnTDL.TAUBnTDLm, delay time is added to positive or negative logic side of the signal (i.e. whether TAUBTTOUTm is set or reset immediately or after dead time has elapsed). The duration of the dead time is specified by slave channel 3.

#### Prerequisites

- Three channels. Select an even channel CH (a) and an odd channel CH (a + 1) for slave channel 2 and 3 respectively.
- The operation mode of the master channel must be set to interval timer mode, see **Table 24.140, Contents of the TAUBnCMORm Register for the Master Channel of the Triangle PWM Output Function with Dead Time**
- Slave channel 1 is not used for this function. This ensures that slave channel 2 is an even-numbered channel (a), and slave channel 3 is an odd-numbered channel (a + 1). Slave channel 1 can be used as a separate timer (independent function).
- The operation mode of slave channel 2 must be set to up down mode, see **Table 24.144, Contents of the TAUBnCMORm Register for the Slave Channel 2 of the Triangle PWM Output Function with Dead Time**
  Furthermore, slave channel 2 must be an even channel
- The operation mode of slave channel 3 must be set to one-count mode, see **Table 24.148, Contents of the TAUBnCMORm Register for the Slave Channel 3 of the Triangle PWM Output Function with Dead Time**
  Furthermore, slave channel 3 must be an odd channel
- The channel output mode of the master channel must be set to independent channel output mode 1
- The channel output mode of the slave channels 2 and 3 must be set to synchronous channel output mode 2.
- The following settings establish TAUBTTOUTm at high level for the down status of the carrier cycle.
  - If the TAUBnCMORm.MD0 (master) bit is set to 0,TAUBnTO.TAUBnTOm must be set to 1 while TAUBnTOE.TAUBnTOEm is 0. (recommended)
  - If the TAUBnCMORm.MD0 (master) bit is set to 1, TAUBnTO.TAUBnTOm must be set to 0 while TAUBnTOE.TAUBnTOEm is 0.

**NOTE**

Slave channel 1 is not used for Triangle PWM Output Function with Dead Time. Slave channel 1 can be used as a separate timer (independent function).

### Functional description

The counters are enabled by setting the channel trigger bits (TAUBnTS.TAUBnTSm) to 1. This in turn sets TAUBnTE.TAUBnTEm = 1, enabling count operation. The current values of TAUBnCDRm is written to TAUBnCNTm and the counters start to count down from these values. If the master channel TAUBnCMORm.TAUBnMD0 bit is set to 1, an interrupt is generated and TAUBTTOUTm signal of the master toggles.

- Master channel:
  When the counter of the master channel reaches $0000_H$, INTTAUBnIm is generated and the TAUBTTOUTm signal toggles. The counter reloads the TAUBnCDRm value and counts down.
- Slave channel 2:
  INTTAUBnIm generated on the master channel triggers the counter of the slave channel 2:
  - If the slave counter currently counts down, it changes count direction.
  - If the slave counter currently counts up, the value of TAUBnCDRm is reloaded and the counter counts down.

  The counter continues to count down or up and awaits the next INTTAUBnIm of the master channel.
  When the counter value of slave channel 2 reaches $0001_H$, INTTAUBnIm is generated.
- Slave channel 3:
  INTTAUBnIm of slave channel 2 triggers the counter of slave channel 3. The current value of TAUBnCDRm (slave 3) is written to TAUBnCNTm (slave 3) and the counter starts to count down from this TAUBnCDRm value.
  When the counter reaches $0000_H$, INTTAUBnIm is generated. The counter returns to $FFFF_H$ and awaits the next INTTAUBnIm of slave channel 2.

The TAUBnTDL.TAUBnTDLm settings of the corresponding channel specify whether it is set/reset immediately, or after dead time has elapsed, as shown in **Table 24.139, Behavior of TAUBTTOUTm when an Interrupt Occurs on Slave Channel 2**.

The TAUBnTOL.TOLm settings specify whether set corresponds to a high signal (TAUBnTOL.TAUBnTOLm = 0) or a low signal (TAUBnTOL.TAUBnTOLm = 1).

The counter can be stopped by setting TAUBnTT.TAUBnTTm to 1 for the master and slave channel(s), which in turn sets TAUBnTE.TAUBnTEm to 0. TAUBnCNTm and TAUBTTOUTm of master and slave channel(s) stop but retain their values.

TAUBnCDRm value of slave channel 2 can be set to $0000_H$ to output 100% TAUBTTOUTm.

**NOTE**

If a forced restart is executed during operation, TAUBTTOUTm is not output as a triangle PWM signal.

**Conditions**

Simultaneous rewrite can be used with this function. Please see **Section 24.6, Simultaneous Rewrite**.

TAUBnTOL.TAUBnTOLm and TAUBnTDL.TAUBnTDLm bits should be set before the counter starts, and slave channels 2 and 3 should have opposite TAUBnTOL.TAUBnTOLm settings or opposite TAUBnTDL.TAUBnTDLm settings.

**Table 24.139 Behavior of TAUBTTOUTm when an Interrupt Occurs on Slave Channel 2**

| TAUBnTDL.TAUBnTDLm | Count Direction of Slave Channel 2 when Interrupt is Generated | TAUBTTOUTm Set/Reset Timing |
|---|---|---|
| 0 | Down | Set after dead time has elapsed |
| | Up | Reset immediately |
| 1 | Down | Set immediately |
| | Up | Reset after dead time has elapsed |

### 24.14.6.2 Equations

Pulse cycle = (TAUBnCDRm (master) + 1) × count clock cycle

$0000_H \leq$ TAUBnCDRm (master) $< FFFF_H$

Carrier cycle (down/up) = (TAUBnCDRm (master) + 1) × 2 × count clock cycle

PWM signal width (positive phase) = [(TAUBnCDRm (master) + 1 – TAUBnCDRm (slave 2)) × 2 – (TAUBnCDRm (slave 3) + 1)] × count clock cycle

PWM signal width (negative phase) = [(TAUBnCDRm (master) + 1 – TAUBnCDRm (slave 2)) × 2 + (TAUBnCDRm (slave 3) + 1)] × count clock cycle

### 24.14.6.3 Block Diagram and General Timing Diagram

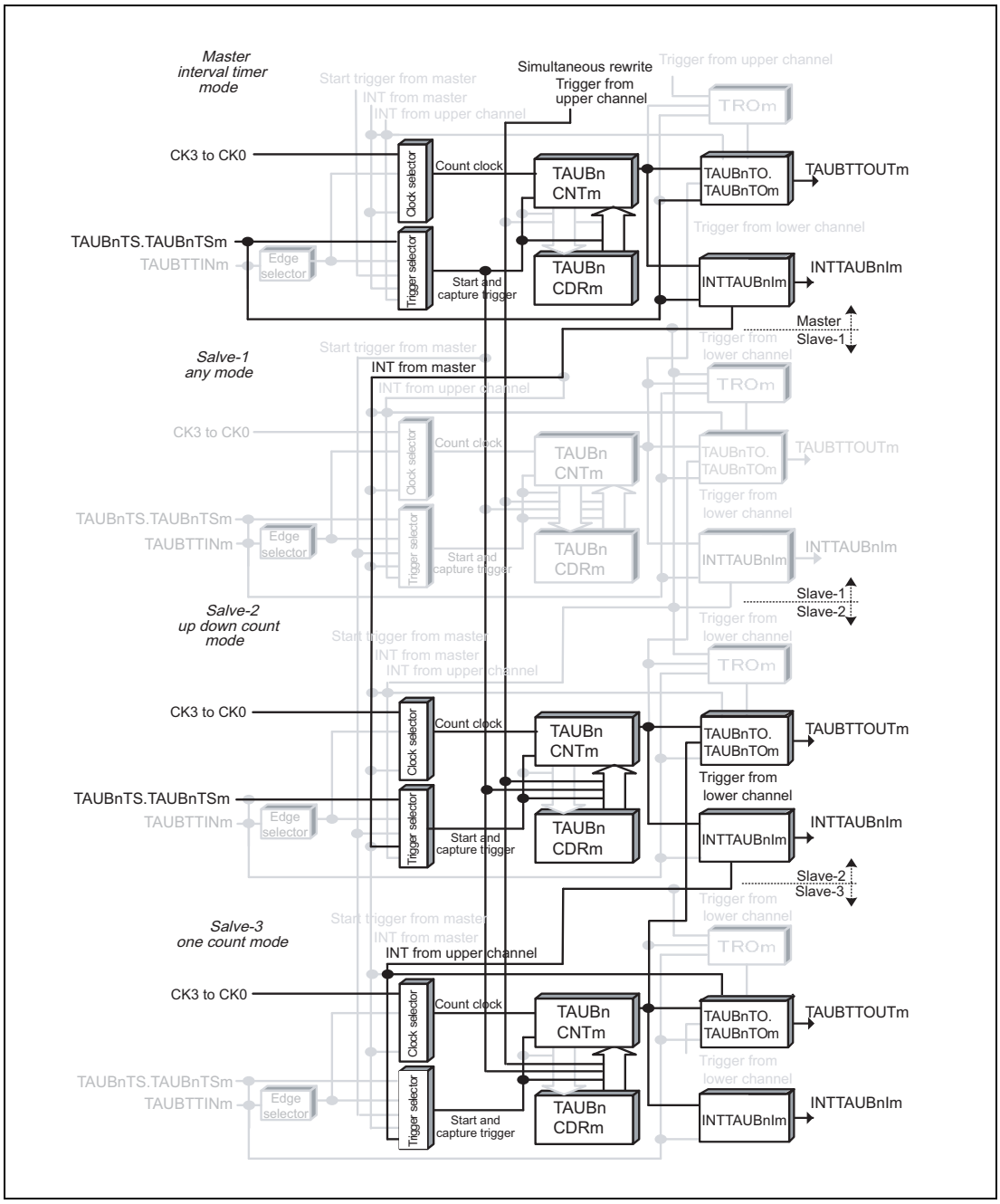



Figure 24.102 Block Diagram for Triangle PWM Output Function with Dead Time

The following settings apply to the general timing diagram.

- Master channel:
  - INTTAUBnIm is generated at operation start (TAUBnCMORm.TAUBnMD0 = 1)
- Slave channel 2:
  - INTTAUBnIm is not generated at operation start (TAUBnCMORm.TAUBnMD0 = 0)
  - TAUBnTDL.TAUBnTDLm = 0
  - Positive logic (TAUBnTOL.TAUBnTOLm = 0)
- Slave channel 3:
  - Enables start trigger detection during counting (TAUBnCMORm.TAUBnMD0 = 1)
  - TAUBnTDL.TAUBnTDLm = 1
  - Positive logic (TAUBnTOL.TAUBnTOLm = 0)

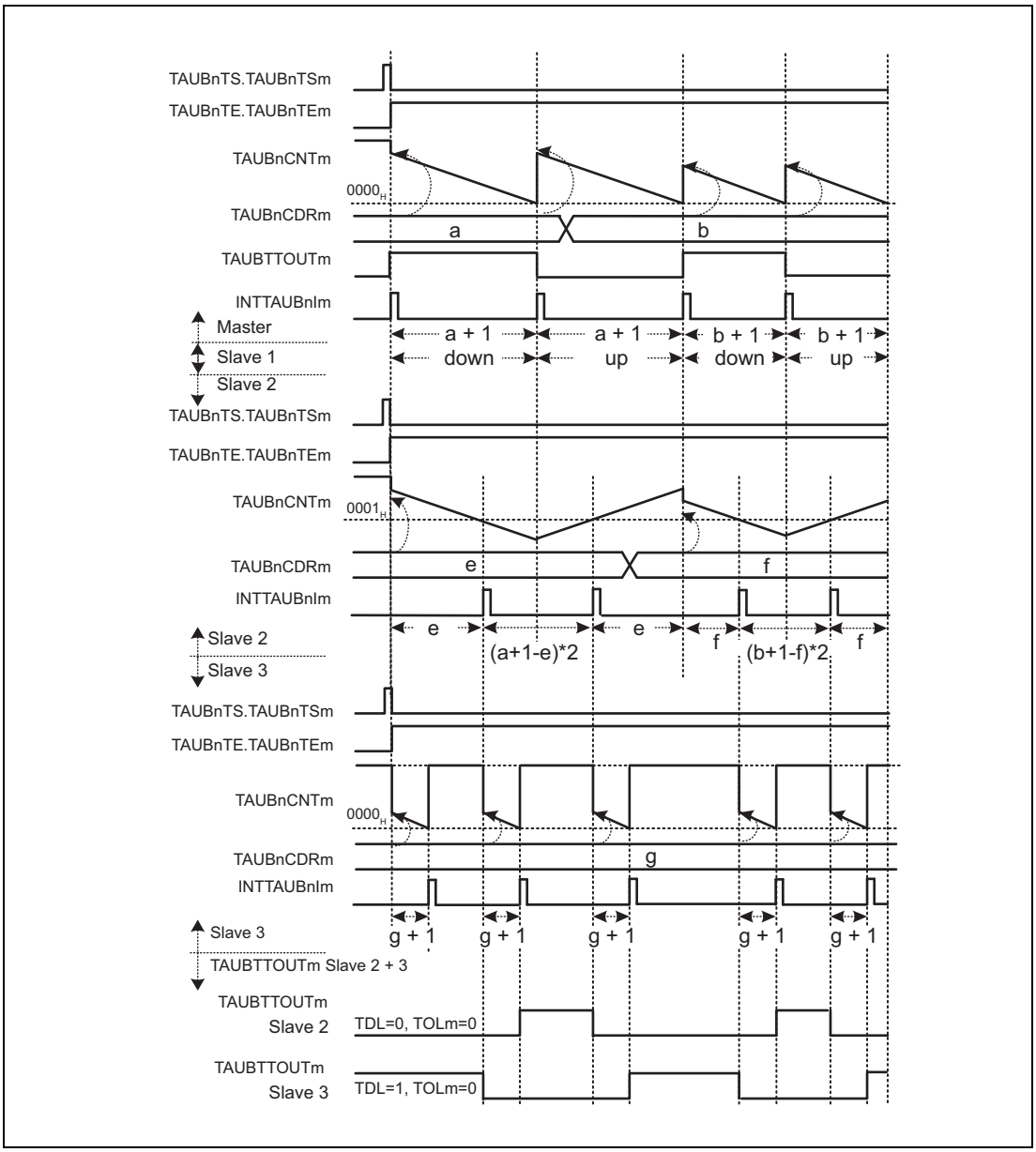



**Figure 24.103 General Timing Diagram for Triangle PWM Output Function with Dead Time**

### 24.14.6.4 Register Settings for the Master Channel

#### (1) TAUBnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.140 Contents of the TAUBnCMORm Register for the Master Channel of the Triangle PWM Output Function with Dead Time**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUBnCKS[1:0] bits of the master and slave channel(s) must be identical. |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $1_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $000_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0000_B$. |
| 0 | TAUBnMD0 | 0: INTTAUBnIm not generated and TAUBTTOUTm does not toggle at operation start<br>1: Generates INTTAUBnIm and toggles TAUBTTOUTm at operation start |

#### (2) TAUBnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.141 Contents of the TAUBnCMURm Register for the Master Channel of the Triangle PWM Output Function with Dead Time**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Not used, so set to 00. |

### (3) Channel output mode for the master channel

Table 24.142 Control Bit Settings for Independent Channel Output Mode 1

| Bit Name | Setting |
|---|---|
| TAUBnTOE.TAUBnTOEm | Write $1_B$. |
| TAUBnTOM.TAUBnTOMm | Write $0_B$. |
| TAUBnTOC.TAUBnTOCm | Write $0_B$. |
| TAUBnTOL.TAUBnTOLm | Write $0_B$. |
| TAUBnTDE.TAUBnTDEm | Write $0_B$. |
| TAUBnTDL.TAUBnTDLm | Write $0_B$. |

### (4) Simultaneous rewrite for the master channel

The simultaneous rewrite settings of the master and slave channels must be identical.

Table 24.143 Simultaneous Rewrite Settings for the Master Channel of the Triangle PWM Output Function with Dead Time

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 1: Enables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: Selects master channel for simultaneous rewrite triggers.<br>1: Selects upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUBnRDM.TAUBnRDMm | 1: The simultaneous rewrite trigger signal is generated when the master channel starts counting and the corresponding slave channel is at the peak of a triangular wave |
| TAUBnRDC.TAUBnRDCm | 0: Does not use the channel to generate the simultaneous rewrite trigger. |

**NOTE**

If TAUBnRDS.TAUBnRDSm = 1, the master channel requires an upper channel that generates the simultaneous rewrite trigger signal.

### 24.14.6.5 Register Settings for Slave Channel 2

#### (1) TAUBnCMORm for slave channel 2

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.144 Contents of the TAUBnCMORm Register for the Slave Channel 2 of the Triangle PWM Output Function with Dead Time**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUBnCKS[1:0] bits of the master and slave channel(s) must be identical. |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $111_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $1001_B$. |
| 0 | TAUBnMD0 | Write $0_B$. |

#### (2) TAUBnCMURm for slave channel 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.145 Contents of the TAUBnCMURm Register for the Slave Channel 2 of the Triangle PWM Output Function with Dead Time**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Not used, so set to 00. |

### (3) Channel output mode for slave channel 2

Table 24.146 Control Bit Settings for Synchronous Channel Output Mode 2 with Dead Time Output

| Bit Name | Setting |
|---|---|
| TAUBnTOE.TAUBnTOEm | Write $1_B$. |
| TAUBnTOM.TAUBnTOMm | Write $1_B$. |
| TAUBnTOC.TAUBnTOCm | Write $1_B$. |
| TAUBnTOL.TAUBnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUBnTDE.TAUBnTDEm | Write $1_B$. |
| TAUBnTDL.TAUBnTDLm | 0: Dead time is added to the positive phase<br>1: Dead time is added to the negative phase |

**CAUTION**

Set TAUBnTDL.TAUBnTDLm exclusively to the odd channel.

### (4) Simultaneous rewrite for slave channel 2

The simultaneous rewrite settings of the master and slave channels must be identical.

Table 24.147 Simultaneous Rewrite Settings for Slave Channel 2 of the Triangle PWM Output Function with Dead Time

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 1: Enables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: Selects master channel for simultaneous rewrite triggers.<br>1: Selects upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUBnRDM.TAUBnRDMm | 1: The simultaneous rewrite trigger signal is generated when the master channel starts counting and the corresponding slave channel is at the peak of a triangular wave |
| TAUBnRDC.TAUBnRDCm | 0: Does not use the channel to generate the simultaneous rewrite trigger. |

### 24.14.6.6 Register Settings for Slave Channel 3

#### (1) TAUBnCMORm for slave channel 3

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUBnCKS [1:0] | | — | TAUBn CCS0 | TAUBn MAS | TAUBnSTS[2:0] | | | TAUBnCOS [1:0] | | — | TAUBnMD[4:1] | | | | TAUBn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 24.148 Contents of the TAUBnCMORm Register for the Slave Channel 3 of the Triangle PWM Output Function with Dead Time**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUBnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUBnCKS[1:0] bits of the master and slave channel(s) must be identical. |
| 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | TAUBnCCS0 | Write $0_B$. |
| 11 | TAUBnMAS | Write $0_B$. |
| 10 to 8 | TAUBnSTS[2:0] | Write $110_B$. |
| 7, 6 | TAUBnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUBnMD[4:1] | Write $0100_B$. |
| 0 | TAUBnMD0 | Write $1_B$. |

#### (2) TAUBnCMURm for slave channel 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUBnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 24.149 Contents of the TAUBnCMURm Register for the Slave Channel 3 of the Triangle PWM Output Function with Dead Time**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUBnTIS[1:0] | 00: Not used, so set to 00. |

### (3) Channel output mode for slave channel 3

Table 24.150 Control Bit Settings for Synchronous Channel Output Mode 2 with Dead Time Output

| Bit Name | Setting |
|---|---|
| TAUBnTOE.TAUBnTOEm | Write $1_B$. |
| TAUBnTOM.TAUBnTOMm | Write $1_B$. |
| TAUBnTOC.TAUBnTOCm | Write $1_B$. |
| TAUBnTOL.TAUBnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUBnTDE.TAUBnTDEm | Write $1_B$. |
| TAUBnTDL.TAUBnTDLm | 0: Dead time is added to the positive phase<br>1: Dead time is added to the negative phase |

**CAUTION**

Set TAUBnTDL.TAUBnTDLm exclusively to the even channel.

### (4) Simultaneous rewrite for slave channel 3

The simultaneous rewrite settings of the master and slave channels must be identical.

Table 24.151 Simultaneous Rewrite Settings for Slave Channel 3 of the Triangle PWM Output Function with Dead Time

| Bit Name | Setting |
|---|---|
| TAUBnRDE.TAUBnRDEm | 1: Enables simultaneous rewrite |
| TAUBnRDS.TAUBnRDSm | 0: Selects master channel for simultaneous rewrite triggers.<br>1: Selects upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUBnRDM.TAUBnRDMm | 1: The simultaneous rewrite trigger signal is generated when the master channel starts counting and the corresponding slave channel is at the peak of a triangular wave |
| TAUBnRDC.TAUBnRDCm | 0: Does not use the channel to generate the simultaneous rewrite trigger. |

### 24.14.6.7 Operating Procedure for Triangle PWM Output Function with Dead Time

Table 24.152 Operating Procedure for Triangle PWM Output with Dead Time

| | Operation | Status of TAUBn |
|---|---|---|
| Initial channel setting | Master channel: set the TAUBnCMORm and TAUBnCMURm registers and the channel output mode as described in **Section 24.14.6.4, Register Settings for the Master Channel**<br><br>Slave channel 2: set the TAUBnCMORm and TAUBnCMURm registers and the channel output mode as described in **Section 24.14.6.5, Register Settings for Slave Channel 2**<br><br>Slave channel 3: set the TAUBnCMORm and TAUBnCMURm registers and the channel output mode as described in **Section 24.14.6.6, Register Settings for Slave Channel 3**<br><br>Set the values of the TAUBnCDRm registers of all channels | Channel operation is stopped. |
| Start operation | Set TAUBnTS.TAUBnTSm of the master and slave channels to 1 simultaneously.<br>TAUBnTS.TAUBnTSm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm (master and slave channels) is set to 1 and the counters of the master and slave channels start.<br>INTTAUBnIm (master) is generated when TAUBnCMORm.TAUBnMD0 is set to 1 on the master channel. |
| During operation | TAUBnCDRm can be changed at any time.<br><br>TAUBnCNTm and TAUBnRSF.TAUBnRSFm can be read at any time.<br><br>TAUBnRDT.TAUBnRDTm can be changed during operation. | TAUBnCNTm of the master channel and slave channel 2 load TAUBnCDRm and count down.<br>When the counter of the master channel reaches $0000_H$:<br>• INTTAUBnIm (master) is generated<br>• TAUBnCNTm (master) reloads the TAUBnCDRm value and continues count operation<br>• TAUBnCNTm (slave 2) reloads the TAUBnCDRm value or counts in the reverse direction<br>When TAUBnCNTm (slave 2) reaches $0001_H$:<br>• INTTAUBnIm (slave 2) is generated<br>• TAUBnCNTm of slave channel 3 loads the TAUBnCDRm value and counts down<br>When TAUBnCNTm of slave channel 3 = $0000_H$:<br>• INTTAUBnIm is generated |
| Stop operation | Set TAUBnTT.TAUBnTTm of the master and slave channels to 1 simultaneously.<br>TAUBnTT.TAUBnTTm is a trigger bit, so it is automatically cleared to 0. | TAUBnTE.TAUBnTEm is cleared to 0 and the counter stops.<br>TAUBnCNTm and TAUBTTOUTm stop and retain their current values. |

Restart operation

### 24.14.6.8 Specific Timing Diagrams

#### (1) Duty cycle = 0%

The following settings apply to the diagram below.

- Slave channel 2:
  - Positive logic (TAUBnTOL.TAUBnTOLm = 0)
- Slave channel 3:
  - Negative logic (TAUBnTOL.TAUBnTOLm = 1)

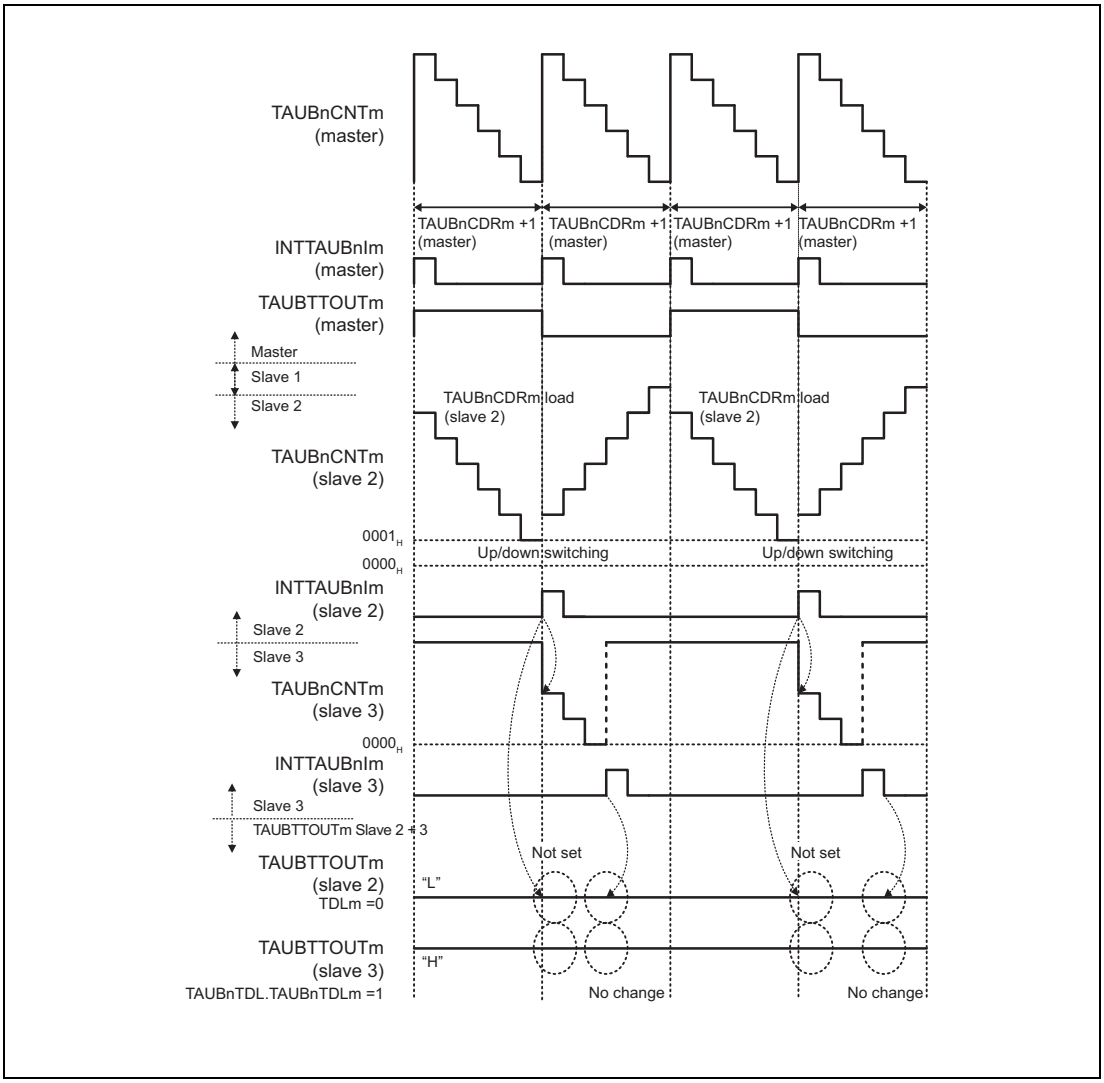



**Figure 24.104 TAUBnCDRm (slave 2) ≥ TAUBnCDRm (master) + 1**

- If TAUBnCDRm (slave 2) ≥ TAUBnCDRm (master), the counter of slave channel cannot reach $0000_H$ during counting down. Therefore TAUBTTOUTm cannot toggle, i.e. it remains at its initial state. The interrupt from slave channel 2 occurs during count up, therefore it is a reset signal.

### (2) Duty cycle = 100%

The following settings apply to the diagram below.

- Slave channel 2:
  - Positive logic (TAUBnTOL.TAUBnTOLm = 0)
- Slave channel 3:
  - Negative logic (TAUBnTOL.TAUBnTOLm = 1)

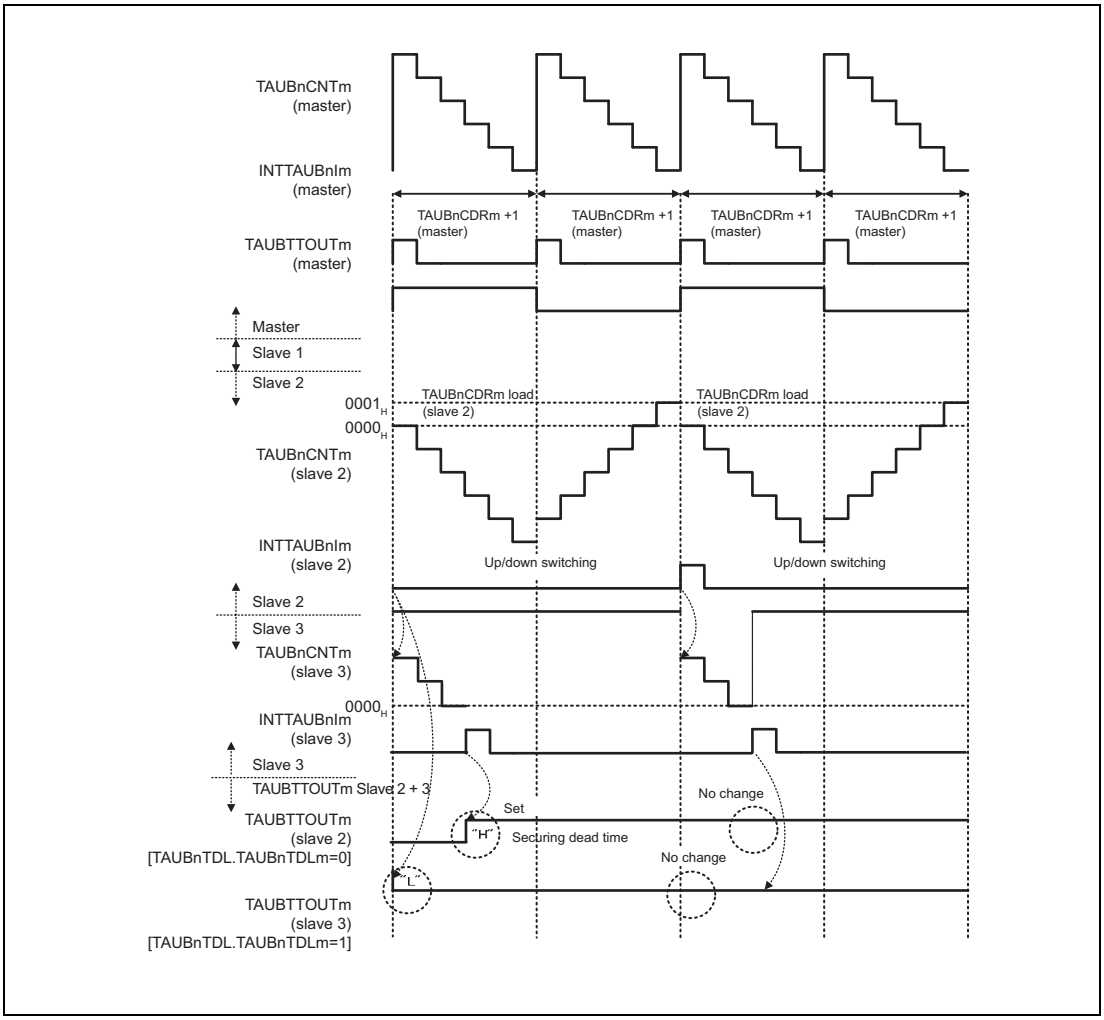



**Figure 24.105 TAUBnCDRm (slave 2) = $0000_H$**

- If TAUBnCDRm (slave 2) = $0000_H$ the counter of slave channel cannot reach $0001_H$ while counting up and therefore cannot generate an INTTAUBnIm while counting up.
  - The set conditions for a channel in which TAUBnTDL.TAUBnTDLm = 0 are met after dead time has elapsed. TAUBTTOUTm toggles but remains in the new state because the reset conditions are never satisfied for such a channel.
  - Slave channel 3 in the above diagram is set when the counter starts.
    However, the reset conditions for a channel in which TAUBnTDL.TAUBnTDLm = 1 are never satisfied so TAUBTTOUTm remains in its initial state for such a slave channel.

### (3) TAUBTTOUTm (slave 2) = 0% and TAUBTTOUTm (slave 3) ≥ 0%

The following settings apply to the diagram below.

- Slave channel 2:
  - Positive logic (TAUBnTOL.TAUBnTOLm = 0)
- Slave channel 3:
  - Negative logic (TAUBnTOL.TAUBnTOLm = 1)

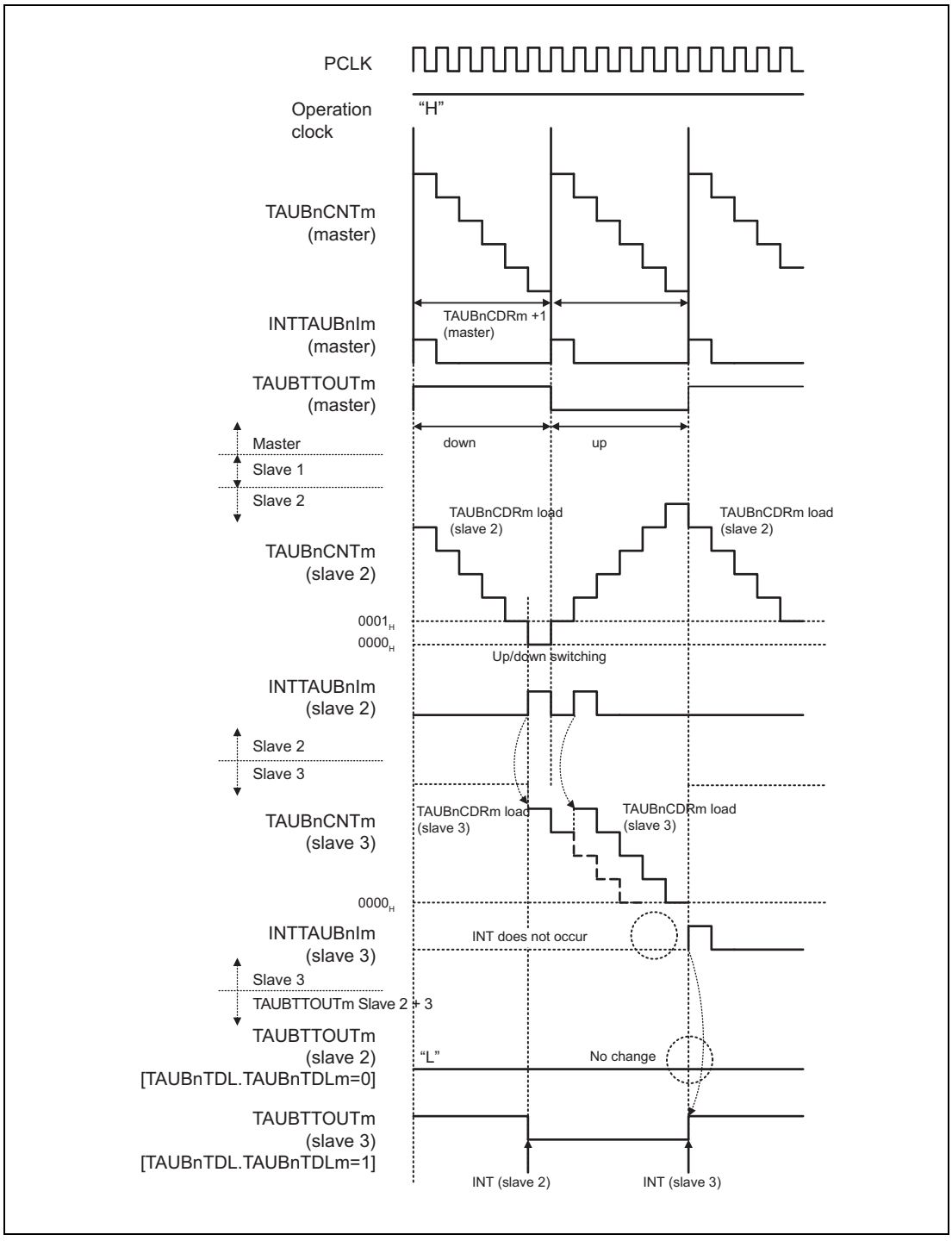



**Figure 24.106 TAUBnCDRm (master) = $0005_H$, TAUBnCDRm (slave 2) = $0005_H$ TAUBnCDRm (slave 3) = $0004_H$**

- When the counter of slave channel 2 reaches $0000_H$ after detecting that the counter reached $0001_H$, INTTAUBnIm (slave 2) is generated. The counter of slave channel 3 starts to count down.
- If another INTTAUBnIm (slave 2) is generated while the counter of slave channel 3 is still counting down, the value of TAUBnCDRm (slave 3) is reloaded and the counter restarts counting down from this value.
- In the diagram above, the first interrupt on channel 2 occurs while the counter is counting down, and the second while it is counting up.
- After the first interrupt, a slave for which TAUBnTDL.TAUBnTDLm = 0 waits for dead time to elapse before setting. However, if another interrupt occurs on slave 2, before the dead time has elapsed, the counter is counting up, so the signal acts as a reset signal, meaning that a channel for which TAUBnTDL.TAUBnTDLm = 0 always remains inactive.
- TAUBTTOUTm of a slave channel for which TAUBnTDL.TAUBnTDLm = 1 is set and reset as normal when the corresponding INTTAUBnIm is generated.

### (4) TAUBTTOUTm (slave 2) > 0% and TAUBTTOUTm (slave 3) = 100%

The following settings apply to the diagram below.

- Slave channel 2:
  - Positive logic (TAUBnTOL.TAUBnTOLm = 0)
- Slave channel 3:
  - Negative logic (TAUBnTOL.TAUBnTOLm = 1)

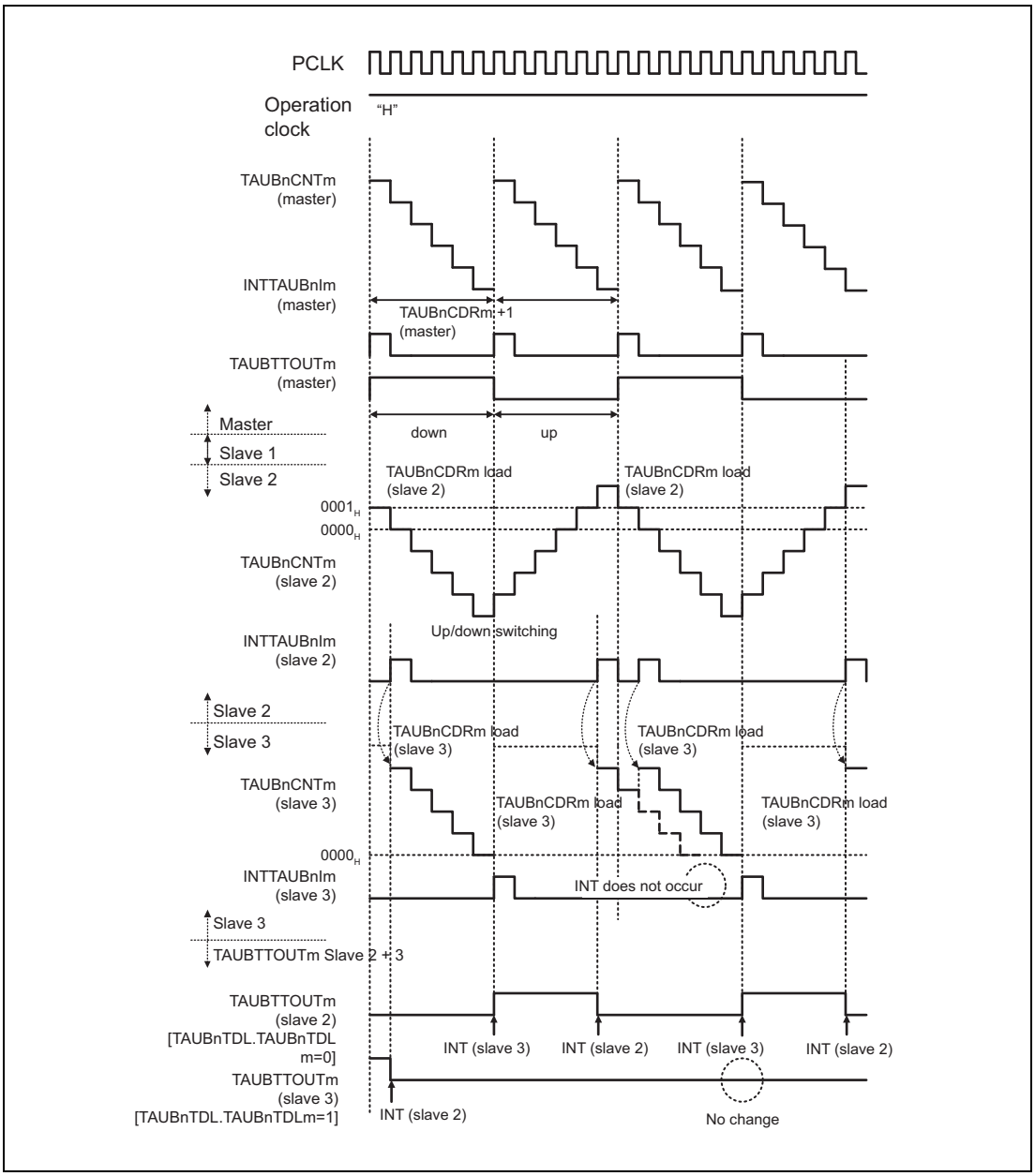



**Figure 24.107 TAUBnCDRm (master) = $0005_H$, TAUBnCDRm (slave 2) = $0001_H$ TAUBnCDRm (slave 3) = $0004_H$ PWM Signal Width (negative phase) ≥ Carrier Cycle**

- After the second interrupt on slave channel 2, a slave for which TAUBnTDL.TAUBnTDLm = 1 is reset after the dead time has elapsed. However if another interrupt occurs on slave 2 before the dead time has elapsed, slave channel 3 is restarted, and then if an interrupt on slave channel 3 is generated, the counter is counting up, so the signal acts as a setting signal, meaning that a channel for which TAUBnTDL.TAUBnTDLm = 1 always remains active.
- TAUBTTOUTm of a slave channel for which TAUBnTDL.TAUBnTDLm = 0 is set and reset as normal when the corresponding INTTAUBnIm is generated.

#### (5) Inhibited INTTAUBnIm to set TAUBTTOUTm positive phase period

The following settings apply to the diagram below.

- Slave channel 2:
  - Positive logic (TAUBnTOL.TAUBnTOLm = 0)
- Slave channel 3:
  - Negative logic (TAUBnTOL.TAUBnTOLm = 1)

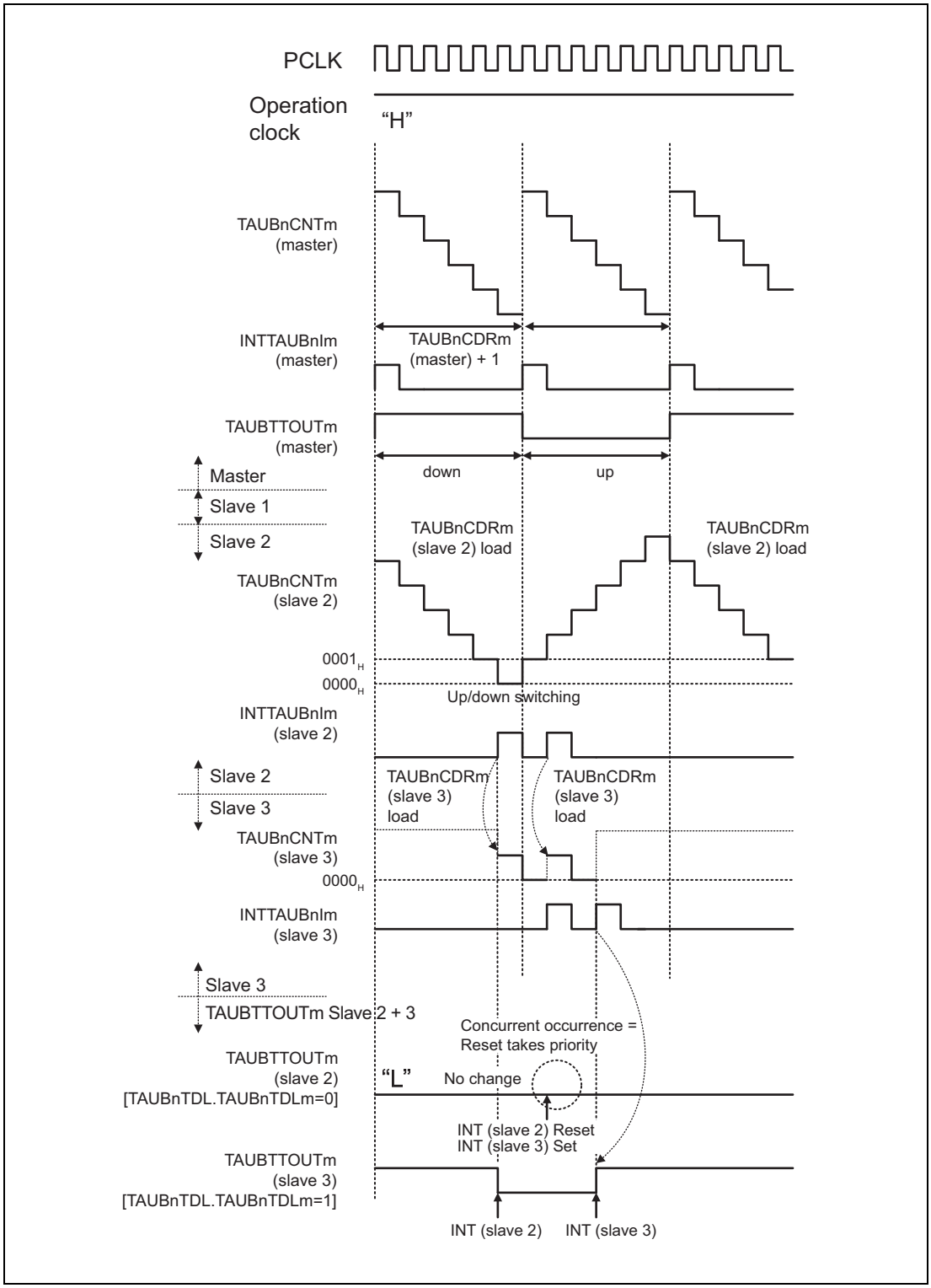



**Figure 24.108 TAUBnCDRm (master) = $0005_H$, TAUBnCDRm (slave 2) = $0005_H$, TAUBnCDRm (slave 3) = $0001_H$ PWM Signal Width (positive phase) = 0**

- The counter of slave channel 3 reaches $0000_H$ and generates an INTTAUBnIm to set the TAUBTTOUTm of slave channel for which TAUBnTDL.TAUBnTDLm = 0 (slave channel 2 in this example).
- If slave channel 2 generates an INTTAUBnIm and resets TAUBTTOUTm simultaneously, this reset signal is given priority if TAUBnTOL.TAUBnTOLm = 0 (if TAUBnTOL.TAUBnTOLm = 1, the set signal is given priority).
- Therefore, TAUBTTOUTm of a slave channel for which TAUBnTDL.TAUBnTDLm = 0 remains in the value after reset.

### (6) Inhibited INTTAUBnIm to set TAUBTTOUTm negative phase period

The following settings apply to the diagram below.

- Slave channel 2:
  - Positive logic (TAUBnTOL.TAUBnTOLm = 0)
- Slave channel 3:
  - Negative logic (TAUBnTOL.TAUBnTOLm = 1)

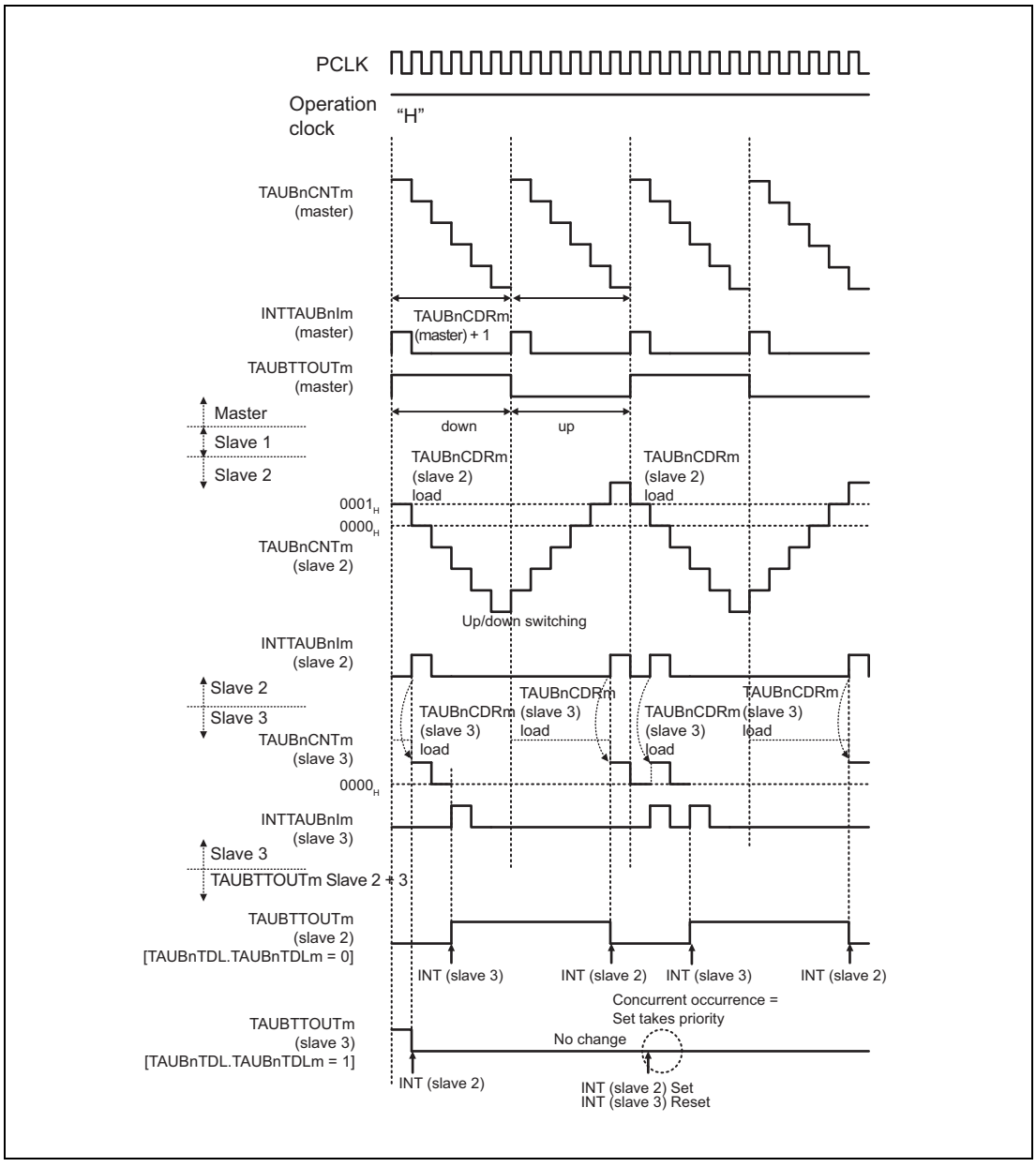



**Figure 24.109 TAUBnCDRm (master) = $0005_H$, TAUBnCDRm (slave 2) = $0001_H$, TAUBnCDRm (slave 3) = $0001_H$ PWM Signal Width (negative phase) = Carrier Cycle**

- The counter of slave channel 3 reaches $0000_H$ and generates an INTTAUBnIm to set the TAUBTTOUTm of slave channel for which TAUBnTDL.TAUBnTDLm = 1 (slave 3 in this example).

- If slave channel 2 generates an INTTAUBnIm and resets TAUBTTOUTm simultaneously, the set signal is given priority if TAUBnTOL.TAUBnTOLm = 1 (if TAUBnTOL.TAUBnTOLm = 0, the reset signal is given priority).
- Therefore, TAUBTTOUTm of a slave channel for which TAUBnTDL.TAUBnTDLm = 1 remains in the value after reset.

### (7) Slave 2 TAUBnCDRm = $0000_H$ (Duty cycle = 100%)

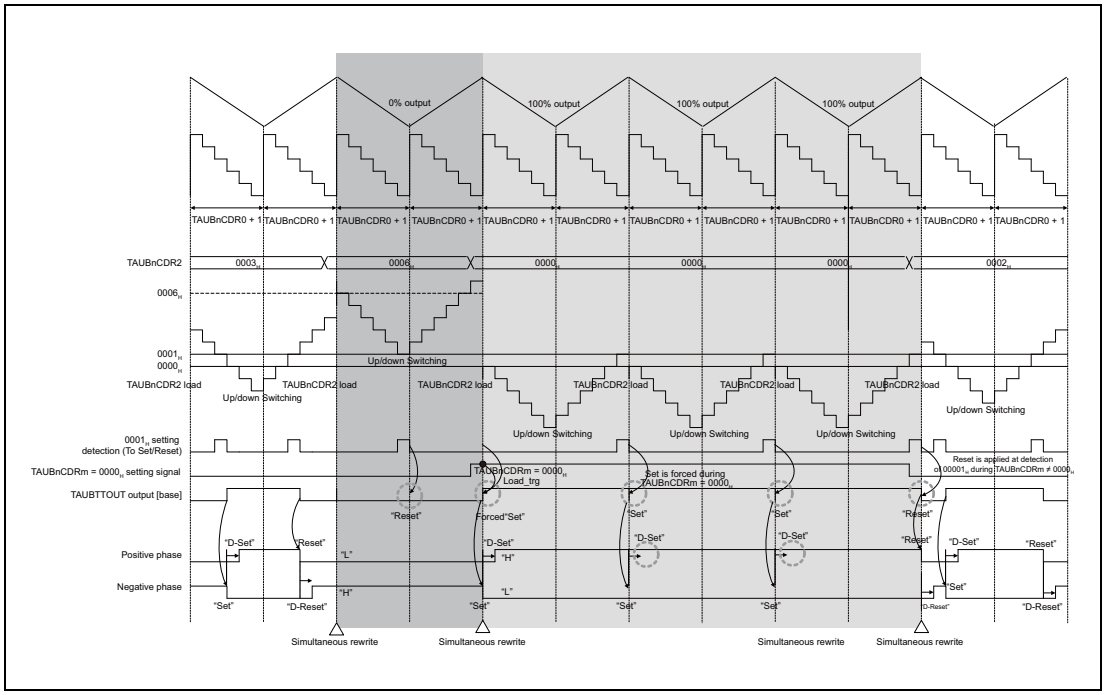



**Figure 24.110 Slave 2 TAUBnCDRm = $0000_H$ (Duty cycle = 100%)**

When rewriting (slave channel 2) TAUBnCDRm ≠ $0000_H$ to (slave channel 2) TAUBnCDRm = $0000_H$ (100% output), set the negative phase side at the start of the carrier cycle, and set the positive phase side after dead time is secured.

When rewriting (slave channel 2) TAUBnCDRm = $0000_H$ (100% output) to (slave channel 2) TAUBnCDRm ≠ $0000_H$, reset the positive phase side at the end of the carrier cycle, and reset the negative phase side after dead time is secured.

## 24.14.7 A/D Conversion Trigger Output Function Type 2

### 24.14.7.1 Overview

#### Summary

This function is identical to **Section 24.14.5, Triangle PWM Output Function**, except that TAUBTTOUTm is not output.

This is achieved by setting the channel output mode of the slave to independent channel output mode controlled by software.

### 24.14.7.2 Block Diagram and General Timing Diagram

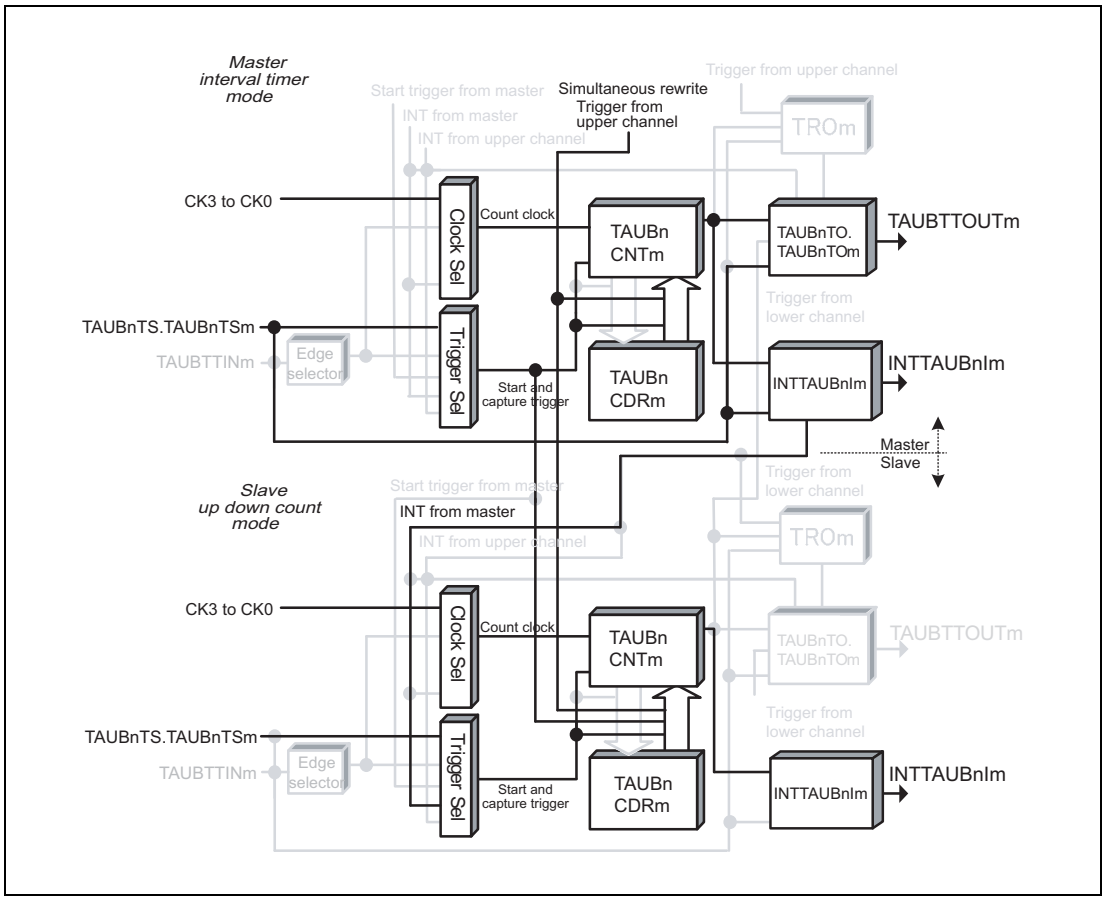



**Figure 24.111 Block Diagram for A/D Conversion Trigger Output Function Type 2**

The following settings apply to the general timing diagram.

- Master channel
  - INTTAUBnIm is generated at operation start (TAUBnCMORm.TAUBnMD0 = 1)

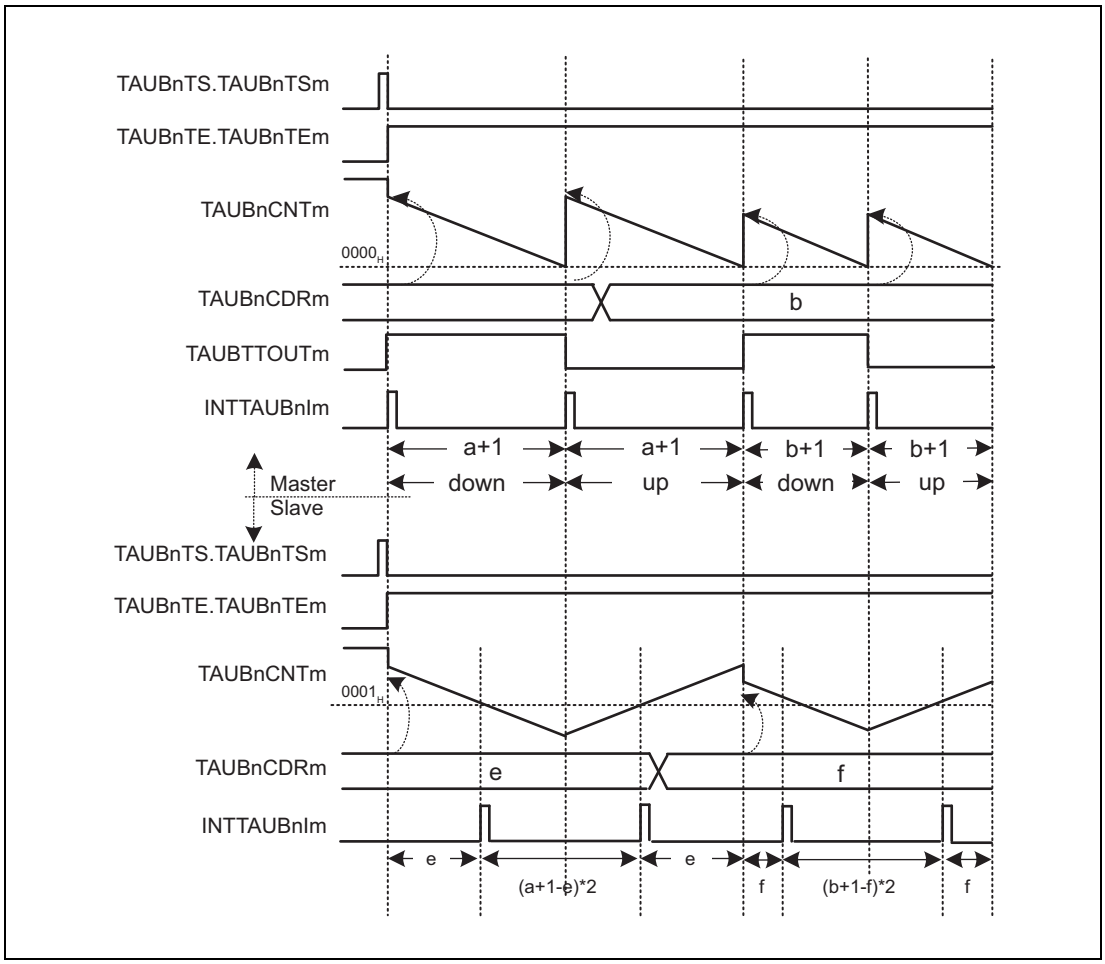



**Figure 24.112 General Timing Diagram for A/D Conversion Trigger Output Function Type 2**

# Section 25 Timer Array Unit D (TAUD)

This section contains a generic description of the timer array unit D (TAUD).

The first part in this section describes the RH850/F1K specific features, such as the number of units and the register base addresses. The remainder of the section describes the functions and registers of the TAUD.

## 25.1 Features of RH850/F1K TAUD

### 25.1.1 Number of Units and Channels

This microcontroller has the following number of TAUD units.

**Table 25.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of Units | 1 | | |
| Name | TAUDn (n = 0) | | |

TAUDn has the following number of channels of timers.

**Table 25.2 TAUDn Unit Configurations and Channels**

| Unit Name (Channel Name) TAUDn | Channels per Unit | RH850/F1K 100 pins (16 ch) | RH850/F1K 144 pins (16 ch) | RH850/F1K 176 pins (16 ch) |
|---|---|---|---|---|
| TAUD0 | 16 | √ | √ | √ |

**Table 25.3 Indices**

| Index | Description |
|---|---|
| n | Throughout this section, the individual TAUD units are identified by the index “n”; for example, TAUDnTOM is the TAUDn channel output mode register. |
| m | The TAUD has 16 channels. Throughout this section, the individual channels are identified by the index “m” (m = 0 to 15), thus a certain channel is denoted as CHm.<br>The even numbered channels (m = 0, 2, 4, 6, 8, 10, 12, 14) are denoted as CHm_even.<br>The odd numbered channels (m = 1, 3, 5, 7, 9, 11, 13, 15) are denoted as CHm_odd. |

### 25.1.2 Register Base Address

TAUDn base addresses are listed in the following table.

TAUDn register addresses are given as offsets from the base addresses.

**Table 25.4 Register Base Address**

| Base Address Name | Base Address |
|---|---|
| <TAUD0_base> | FFE2 $0000_H$ |

### 25.1.3 Clock Supply

The TAUDn clock supply is shown in the following table.

**Table 25.5 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| TAUDn | PCLK | CKSCLK_IPERI1 | Module clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CKSCLK_IPERI1 | |

### 25.1.4 Interrupt Requests

TAUDn interrupt requests are listed in the following table.

**Table 25.6 Interrupt Requests**

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **TAUD0** | | | |
| INTTAUD0I0 | Channel 0 interrupt | 8, 132 | 0 |
| INTTAUD0I1 | Channel 1 interrupt | 48 | 15 |
| INTTAUD0I2 | Channel 2 interrupt | 9, 158 | 64 |
| INTTAUD0I3 | Channel 3 interrupt | 49 | 76 |
| INTTAUD0I4 | Channel 4 interrupt | 10, 133 | 1 |
| INTTAUD0I5 | Channel 5 interrupt | 50 | 16 |
| INTTAUD0I6 | Channel 6 interrupt | 11, 134 | 65 |
| INTTAUD0I7 | Channel 7 interrupt | 51 | 77 |
| INTTAUD0I8 | Channel 8 interrupt | 12, 135 | 2 |
| INTTAUD0I9 | Channel 9 interrupt | 52 | 17 |
| INTTAUD0I10 | Channel 10 interrupt | 13, 159 | 66 |
| INTTAUD0I11 | Channel 11 interrupt | 53 | 78 |
| INTTAUD0I12 | Channel 12 interrupt | 14, 160 | 3 |
| INTTAUD0I13 | Channel 13 interrupt | 54 | 18 |
| INTTAUD0I14 | Channel 14 interrupt | 15, 161 | 67 |
| INTTAUD0I15 | Channel 15 interrupt | 55 | 79 |

### 25.1.5 Reset Sources

TAUDn reset sources are listed in the following table. TAUDn is initialized by these reset sources.

**Table 25.7 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| TAUDn | All reset sources (ISORES) |

### 25.1.6 External Input/Output Signals

External input/output signals of TAUDn are listed below. .

**Table 25.8 External Input/Output Signals**

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **TAUD0** | | |
| TAUDTTIN0 | Channel 0 input*1 | TAUD0I0 |
| TAUDTTIN1 | Channel 1 input*1 | TAUD0I1 |
| TAUDTTIN2 | Channel 2 input*1 | TAUD0I2 |
| TAUDTTIN3 | Channel 3 input*1 | TAUD0I3 |
| TAUDTTIN4 | Channel 4 input*1 | TAUD0I4 |
| TAUDTTIN5 | Channel 5 input*1 | TAUD0I5 |
| TAUDTTIN6 | Channel 6 input*1 | TAUD0I6 |
| TAUDTTIN7 | Channel 7 input*1 | TAUD0I7 |
| TAUDTTIN8 | Channel 8 input*1 | TAUD0I8 |
| TAUDTTIN9 | Channel 9 input*1 | TAUD0I9 |
| TAUDTTIN10 | Channel 10 input*1 | TAUD0I10 |
| TAUDTTIN11 | Channel 11 input*1 | TAUD0I11 |
| TAUDTTIN12 | Channel 12 input*1 | TAUD0I12 |
| TAUDTTIN13 | Channel 13 input*1 | TAUD0I13 |
| TAUDTTIN14 | Channel 14 input*1 | TAUD0I14 |
| TAUDTTIN15 | Channel 15 input*1 | TAUD0I15 |
| TAUDTTOUT0 | Channel 0 output | TAUD0O0 |
| TAUDTTOUT1 | Channel 1 output | TAUD0O1 |
| TAUDTTOUT2 | Channel 2 output | TAUD0O2 |
| TAUDTTOUT3 | Channel 3 output | TAUD0O3 |
| TAUDTTOUT4 | Channel 4 output | TAUD0O4 |
| TAUDTTOUT5 | Channel 5 output | TAUD0O5 |
| TAUDTTOUT6 | Channel 6 output | TAUD0O6 |
| TAUDTTOUT7 | Channel 7 output | TAUD0O7 |
| TAUDTTOUT8 | Channel 8 output | TAUD0O8 |
| TAUDTTOUT9 | Channel 9 output | TAUD0O9 |
| TAUDTTOUT10 | Channel 10 output | TAUD0O10 |
| TAUDTTOUT11 | Channel 11 output | TAUD0O11 |
| TAUDTTOUT12 | Channel 12 output | TAUD0O12 |
| TAUDTTOUT13 | Channel 13 output | TAUD0O13 |
| TAUDTTOUT14 | Channel 14 output | TAUD0O14 |
| TAUDTTOUT15 | Channel 15 output | TAUD0O15 |

Note 1. When channel input pins are to be used, noise filters must be set for the corresponding port pin functions. For details, see **Section 2.12, Noise Filter & Edge/Level Detector**.

## 25.1.7 Internal Input/Output Signals

The internal input/output signals of TAUDn are listed below.

**Table 25.9 Internal Input/Output Signals**

| Unit Signal Name | Description | Connected to |
|---|---|---|
| TAUDnTSSTm | Simultaneous channel start trigger input | PIC |
| TAUDnTUDCm (m = 0, 2, 8) | TAUD master up/down signal output | PIC |

## 25.1.8 TAUD0 Input Selection

The output from port TAUD0Im (m = 0 to 15) can be input to TAUDTTINm (m = 0 to 15) as shown in the following figure.

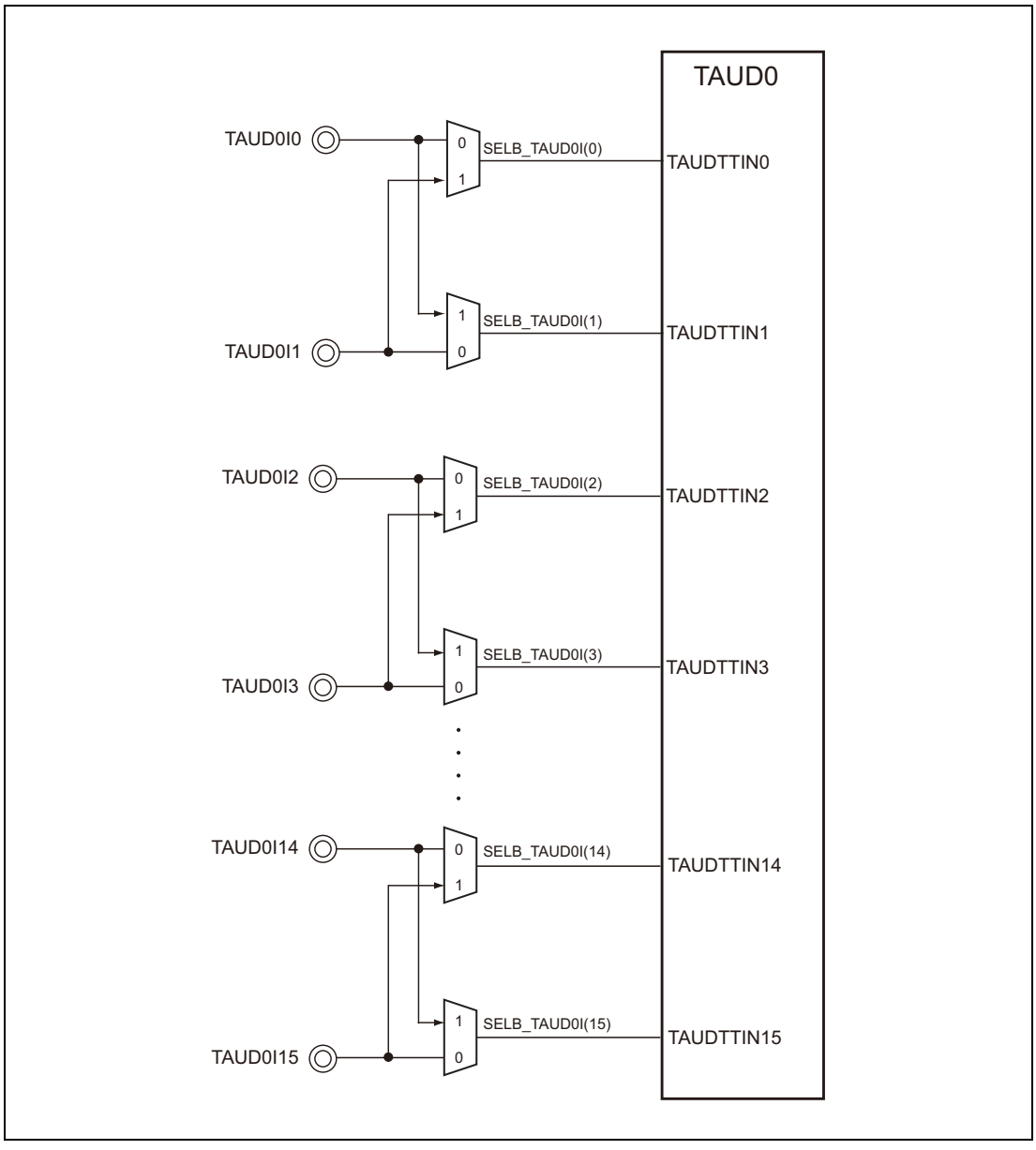



**Figure 25.1 Selection of Signals Input to TAUD0**

The following table shows the method of selecting input signals to several TAUD0 inputs.

**Table 25.10 TAUD0 Input Selection**

| Input Signal | Function | Settings |
|---|---|---|
| TAUDTTIN [m] | Port TAUD0I[m] | SELB_TAUD0I [m] = 0 |
| | Port TAUD0I[m + 1] | SELB_TAUD0I [m] = 1 |
| TAUDTTIN [m + 1] | Port TAUD0I[m + 1] | SELB_TAUD0I [m + 1] = 0 |
| | Port TAUD0I[m] | SELB_TAUD0I [m + 1] = 1 |

### 25.1.8.1 List of Registers

Input signal selection register is listed in the following table.

**Table 25.11 List of Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| SL_TAUD0 | TAUDTTINm Input Signal Selection Register | SELB_TAUD0I | FFE2 $4000_H$ |

### 25.1.8.2 SELB_TAUD0I — TAUDTTINm Input Signal Selection Register

This register selects the TAUDTTINm input signals.

**Access:** This register can be read or written in 16-bit units.

**Address:** FFE2 $4000_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SELB_TAUD0I15 | SELB_TAUD0I14 | SELB_TAUD0I13 | SELB_TAUD0I12 | SELB_TAUD0I11 | SELB_TAUD0I10 | SELB_TAUD0I9 | SELB_TAUD0I8 | SELB_TAUD0I7 | SELB_TAUD0I6 | SELB_TAUD0I5 | SELB_TAUD0I4 | SELB_TAUD0I3 | SELB_TAUD0I2 | SELB_TAUD0I1 | SELB_TAUD0I0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.12 SELB_TAUD0I Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | SELB_TAUD0Im | Selection of TAUDTTINm input signal |

| TAUD Input | Bit [m+1] | Bit [m] | Input signal |
|---|---|---|---|
| TAUDTTIN[m] | x | 0 | Selection of port TAUD0I[m] |
| | x | 1 | Selection of port TAUD0I[m + 1] |
| TAUDTTIN[m+1] | 0 | x | Selection of port TAUD0I[m + 1] |
| | 1 | x | Selection of port TAUD0I[m] |

(m = 0, 2, 4, 6, 8, 10, 12, 14)

**CAUTION**

Do not change the input signal of each channel during the timer counting.

## 25.2 Overview

### 25.2.1 Functional Overview

The TAUD has the following functions:

- 16 channels
- 16-bit counter and 16-bit data register per channel
- Independent channel operation
- Synchronous channel operation (master and slave operation)
- Generation of different types of output signal
- Real-time output
- Counter can be triggered by external signal
- Interrupt generation

The Timer Array Unit D is used to perform various count or timer operations and to output a signal which depends on the result of the operation. It contains one prescaler block for count clock generation and 16 channels, each equipped with a 16-bit counter TAUDnCNTm and a 16-bit data register TAUDnCDRm to hold the start or compare value of the counter.

It also contains several control and status registers.

#### Independent and synchronous operation

Every channel can operate in different operation modes, either independently or in combination with other channels (synchronously). When one master channel and one or more slave channels operate in combination, the slave channels depend on the master channel.

When a channel is operated independently, its operation mode and functions are not affected by those of other channels. When a channel is operated synchronously it is either a master or a slave. A master channel can have multiple slaves, and the state of one channel affects that of the other channels. For example, this means that one channel can control when another starts to count, is reset, etc.

## 25.2.2 Terms

In this section, the following terms are used.

### Independent / synchronous channel operation

Independent or synchronous channel operation describes the dependency of channels on each other:

- If a channel operates independently of all other channels, this is called independent channel operation.
- If a channel operates depending on other channels, this is called synchronous channel operation.

### Channel group

In synchronous channel operation, all channels that depend on each other are referred to as a “channel group”.

A channel group has one master channel and one or more slave channels.

### Operation mode

An operation mode can be selected for every channel m. The operation mode defines the basic operation and features of a channel.

In synchronous channel operation, every channel in the channel group can operate in a different operation mode.

Examples are “Capture Mode”, “Event Count Mode”, and “Interval Timer Mode”.

### Channel output mode

The channel output mode defines the operation of TAUDTTOUTm

- of a single channel (independent output operation) or
- of all channels in a channel group (synchronous output operation).

Examples are “Independent Channel Output Mode 1” and “Synchronous Channel Output Mode 2 with Dead Time Output”.

### Channel operation function

The channel operation function defines the complete function and all features

- of a single channel (independent channel operation) or
- of all channels in a channel group (synchronous channel operation).

### Upper / lower channel

Depending on the channel number m, a channel with a smaller number or with a larger number is referred to as “upper” or “lower” channel, respectively.

- Upper channel: Channel with a smaller channel number
- Lower channel: Channel with a larger channel number

Example:

For channel 5, channel 3 is an upper channel and channel 9 is a lower channel.

### 25.2.3 Functional List of Timer Operations

This timer provides the following functions by operating each channel independently or by combining multiple channels.

**Table 25.13 Functional List of TAUD Operations**

| Operation Function | Example |
|---|---|
| **Independent Channel Operation Functions** | **Section 25.12** |
| Interval Timer Function | **Section 25.12.1** |
| TAUDTTINm Input Interval Timer Function | **Section 25.12.2** |
| Clock Divide Function | **Section 25.12.3** |
| External Event Count Function | **Section 25.12.4** |
| Delay Count Function | **Section 25.12.5** |
| One-Pulse Output Function | **Section 25.12.6** |
| TAUDTTINm Input Pulse Interval Measurement Function | **Section 25.12.7** |
| TAUDTTINm Input Signal Width Measurement Function | **Section 25.12.8** |
| TAUDTTINm Input Position Detection Function | **Section 25.12.9** |
| TAUDTTINm Input Period Count Detection Function | **Section 25.12.10** |
| TAUDTTINm Input Pulse Interval Judgment Function | **Section 25.12.11** |
| TAUDTTINm Input Signal Width Judgment Function | **Section 25.12.12** |
| Overflow Interrupt Output Function (during TAUDTTINm Width Measurement) | **Section 25.12.13** |
| Overflow Interrupt Output Function (during TAUDTTINm Input Period Count Detection) | **Section 25.12.14** |
| One-Phase PWM Output Function | **Section 25.12.15** |
| **Independent Channel Real-Time Functions** | **Section 25.13** |
| Real-Time Output Function Type 1 | **Section 25.13.1** |
| Real-Time Output Function Type 2 | **Section 25.13.2** |
| **Independent Channel Simultaneous Rewrite Functions** | **Section 25.14** |
| Simultaneous Rewrite Trigger Generation Function Type 1 | **Section 25.14.1** |
| Simultaneous Rewrite Trigger Generation Function Type 2 | **Section 25.14.2** |
| **Synchronous Channel Operation Functions** | **Section 25.15** |
| PWM Output Function | **Section 25.15.1** |
| One-Shot Pulse Output Function | **Section 25.15.2** |
| Trigger Start PWM Output Function | **Section 25.15.3** |
| Delay Pulse Output Function | **Section 25.15.4** |
| Offset Trigger Output Function | **Section 25.15.5** |
| A/D Conversion Trigger Output Function Type 1 | **Section 25.15.6** |
| Triangle PWM Output Function | **Section 25.15.7** |
| Triangle PWM Output Function with Dead Time | **Section 25.15.8** |
| A/D Conversion Trigger Output Function Type 2 | **Section 25.15.9** |
| Interrupt Request Signals Culling Function | **Section 25.15.10** |
| **Synchronous Non-Complementary and Complementary Modulation Output Functions** | **Section 25.16** |
| Non-Complementary Modulation Output Function Type 1 | **Section 25.16.1** |
| Non-Complementary Modulation Output Function Type 2 | **Section 25.16.2** |
| Complementary Modulation Output Function | **Section 25.16.3** |

### 25.2.4 TAUD I/O and Interrupt Request Signals

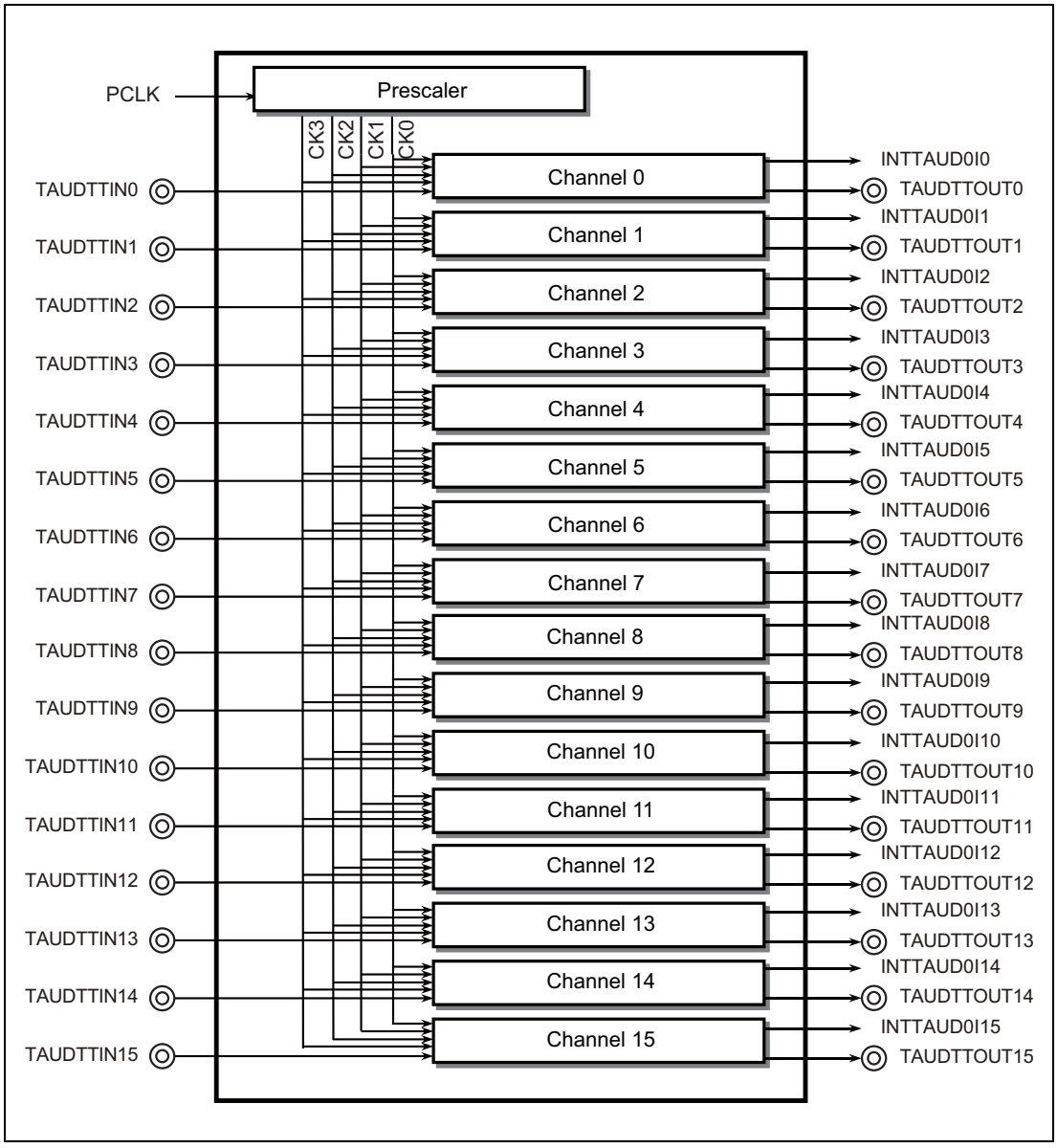



**Figure 25.2 TAUD I/O and Interrupt Request Signals**

## 25.2.5 Block Diagram

**Figure 25.3** shows the main components of the TAUD.

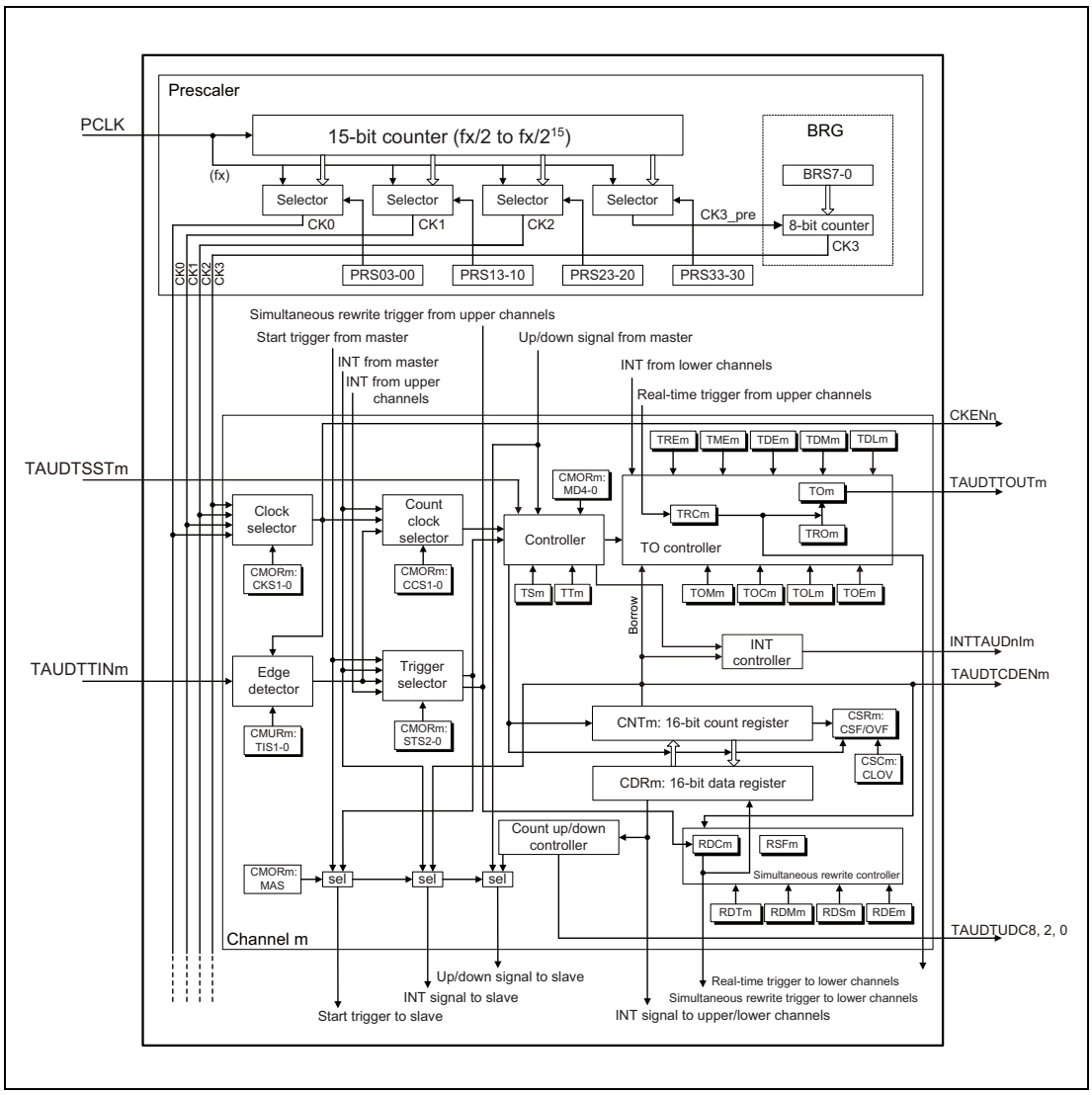



**Figure 25.3 Block Diagram of the TAUD**

The module name “TAUDn” has been omitted from the register names for the sake of clarity in the above figure.

## 25.2.6 Description of Blocks

The following describes the functional blocks:

### Prescaler block

The prescaler block provides up to four clock signals (CK0 to CK3) that can be used as count clocks for all channels.

Count clocks CK0 to CK2 are derived from PCLK by a configurable prescaler division factor of $2^0$ to $2^{15}$. The fourth count clock CK3 can be adjusted more precisely by using BRG to set an additional division factor that is not a power of 2.

### Clock and count clock selection

For every channel, the count clock selector selects which of the following is used as the clock source:

- One of the clocks CK0 to CK3 (selected by the clock selector)
- INTTAUDnIm from master channel
- TAUDTTINm input signal valid edge

### Controller

The controller controls the main operations of the counter:

- Operation mode (selected by bits TAUDnCMORm.TAUDnMD[4:0])
- Counter start enable (TAUDnTS.TAUDnTSm) and counter stop (TAUDnTT.TAUDnTTm)
  When counter start is enabled, status flag TAUDnTE.TAUDnTEm is set.
- Count direction (up/down) (can be controlled by master channel)

### Trigger selector

Depending on the selected operation mode, the counter starts automatically when it is enabled (TAUDnTE.TAUDnTEm = 1), or it waits for an external start trigger signal. Any of the following signals can be used as the start trigger.

- Synchronous channel start trigger input TAUDnTSSTm
- TAUDTTINm input signal valid edge
- INTTAUDnIm from the master or any upper channel
- Up/down output trigger signal of the master channel
- Dead-time output signal of the TAUDTTOUTm generation unit.

### Simultaneous rewrite controller

Simultaneous rewrite control is a function that can be used in synchronous operating modes. The data registers (TAUDnCDRm) of all channels in a channel group can be rewritten at any time. The simultaneous rewrite controller ensures that new data register values of all channels become effective at the same time.

### TAUDnTO controller

The output control of every channel enables the generation of various output signal forms such as PWM signals or triangular waves.

# 25.3 Registers

## 25.3.1 List of Registers

TAUD registers are listed in the following table.

For details about <TAUDn_base>, see **Section 25.1.2, Register Base Address**.

**Table 25.14 List of Registers**

| Module | Register | Symbol | Address |
|---|---|---|---|
| **TAUDn prescaler registers** | | | |
| TAUDn | TAUDn prescaler clock select register | TAUDnTPS | <TAUDn_base> + $240_H$ |
| | TAUDn prescaler baud rate setting register | TAUDnBRS | <TAUDn_base> + $244_H$ |
| **TAUDn control registers** | | | |
| TAUDn | TAUDn channel data register m | TAUDnCDRm | <TAUDn_base> + m × $4_H$ |
| | TAUDn channel counter register m | TAUDnCNTm | <TAUDn_base> + $80_H$ + m × $4_H$ |
| | TAUDn channel mode OS register m | TAUDnCMORm | <TAUDn_base> + $200_H$ + m × $4_H$ |
| | TAUDn channel mode user register m | TAUDnCMURm | <TAUDn_base> + $C0_H$ + m × $4_H$ |
| | TAUDn channel status register m | TAUDnCSRm | <TAUDn_base> + $140_H$ + m × $4_H$ |
| | TAUDn channel status clear trigger register m | TAUDnCSCm | <TAUDn_base> + $180_H$ + m × $4_H$ |
| | TAUDn channel start trigger register | TAUDnTS | <TAUDn_base> + $1C4_H$ |
| | TAUDn channel enable status register | TAUDnTE | <TAUDn_base> + $1C0_H$ |
| | TAUDn channel stop trigger register | TAUDnTT | <TAUDn_base> + $1C8_H$ |
| **TAUDn output registers** | | | |
| TAUDn | TAUDn channel output enable register | TAUDnTOE | <TAUDn_base> + $5C_H$ |
| | TAUDn channel output register | TAUDnTO | <TAUDn_base> + $58_H$ |
| | TAUDn channel output mode register | TAUDnTOM | <TAUDn_base> + $248_H$ |
| | TAUDn channel output configuration register | TAUDnTOC | <TAUDn_base> + $24C_H$ |
| | TAUDn channel output active level register | TAUDnTOL | <TAUDn_base> + $040_H$ |
| | TAUDn channel dead time output enable register | TAUDnTDE | <TAUDn_base> + $250_H$ |
| | TAUDn channel dead time output mode register | TAUDnTDM | <TAUDn_base> + $254_H$ |
| | TAUDn channel dead time output level register | TAUDnTDL | <TAUDn_base> + $54_H$ |
| | TAUDn channel real-time output register | TAUDnTRO | <TAUDn_base> + $4C_H$ |
| | TAUDn channel real-time output enable register | TAUDnTRE | <TAUDn_base> + $258_H$ |
| | TAUDn channel real-time output control register | TAUDnTRC | <TAUDn_base> + $25C_H$ |
| | TAUDn channel modulation output enable register | TAUDnTME | <TAUDn_base> + $50_H$ |
| **TAUDn reload data registers** | | | |
| TAUDn | TAUDn channel reload data enable register | TAUDnRDE | <TAUDn_base> + $260_H$ |
| | TAUDn channel reload data mode register | TAUDnRDM | <TAUDn_base> + $264_H$ |
| | TAUDn channel reload data control CH select register | TAUDnRDS | <TAUDn_base> + $268_H$ |
| | TAUDn channel reload data control register | TAUDnRDC | <TAUDn_base> + $26C_H$ |
| | TAUDn channel reload data trigger register | TAUDnRDT | <TAUDn_base> + $44_H$ |
| | TAUDn channel reload status register | TAUDnRSF | <TAUDn_base> + $48_H$ |
| **TAUDn Emulation Register** | | | |
| TAUDn | TAUDn emulation register | TAUDnEMU | <TAUDn_base> + $290_H$ |

## 25.3.2 Details of TAUDn Prescaler Registers

### 25.3.2.1 TAUDnTPS — TAUDn Prescaler Clock Select Register

This register specifies clocks CK0, CK1, CK2, and CK3_PRE for all channels of the PCLK prescaler. CK3 is generated by dividing CK3_PRE by the factor specified in TAUDnBRS.

**Access:** This register can be read or written in 16-bit units.

**Address:** <TAUDn_base> + $240_H$

**Value after reset:** $FFFF_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnPRS3 [3:0] | | | | TAUDnPRS2 [3:0] | | | | TAUDnPRS1 [3:0] | | | | TAUDnPRS0 [3:0] | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.15 TAUDnTPS Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 12 | TAUDnPRS3 [3:0] | Specifies CK3_PRE clock.<br>CK3_PRE clock is an input clock to BRG unit which supplies the CK3 operation clock to all channels. |

| TAUDnPRS3[3:0] | CK3_PRE Clock |
|---|---|
| $0000_B$ | PCLK/$2^0$ |
| $0001_B$ | PCLK/$2^1$ |
| $0010_B$ | PCLK/$2^2$ |
| $0011_B$ | PCLK/$2^3$ |
| $0100_B$ | PCLK/$2^4$ |
| $0101_B$ | PCLK/$2^5$ |
| $0110_B$ | PCLK/$2^6$ |
| $0111_B$ | PCLK/$2^7$ |
| $1000_B$ | PCLK/$2^8$ |
| $1001_B$ | PCLK/$2^9$ |
| $1010_B$ | PCLK/$2^{10}$ |
| $1011_B$ | PCLK/$2^{11}$ |
| $1100_B$ | PCLK/$2^{12}$ |
| $1101_B$ | PCLK/$2^{13}$ |
| $1110_B$ | PCLK/$2^{14}$ |
| $1111_B$ | PCLK/$2^{15}$ |

The above bits are rewritable only when all the counters using CK3 are stopped (TAUDnTE.TAUDnTEm = 0).

**Table 25.15 TAUDnTPS Register Contents (2/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 to 8 | TAUDnPRS2 [3:0] | Specifies the CK2 clock. |

| TAUDnPRS2[3:0] | CK2 Clock |
|---|---|
| $0000_B$ | $PCLK/2^0$ |
| $0001_B$ | $PCLK/2^1$ |
| $0010_B$ | $PCLK/2^2$ |
| $0011_B$ | $PCLK/2^3$ |
| $0100_B$ | $PCLK/2^4$ |
| $0101_B$ | $PCLK/2^5$ |
| $0110_B$ | $PCLK/2^6$ |
| $0111_B$ | $PCLK/2^7$ |
| $1000_B$ | $PCLK/2^8$ |
| $1001_B$ | $PCLK/2^9$ |
| $1010_B$ | $PCLK/2^{10}$ |
| $1011_B$ | $PCLK/2^{11}$ |
| $1100_B$ | $PCLK/2^{12}$ |
| $1101_B$ | $PCLK/2^{13}$ |
| $1110_B$ | $PCLK/2^{14}$ |
| $1111_B$ | $PCLK/2^{15}$ |

The above bits are rewritable only when all the counters using CK2 are stopped (TAUDnTE.TAUDnTEm = 0).

| | | |
|---|---|---|
| 7 to 4 | TAUDnPRS1 [3:0] | Specifies the CK1 clock. |

| TAUDnPRS1[3:0] | CK1 Clock |
|---|---|
| $0000_B$ | $PCLK/2^0$ |
| $0001_B$ | $PCLK/2^1$ |
| $0010_B$ | $PCLK/2^2$ |
| $0011_B$ | $PCLK/2^3$ |
| $0100_B$ | $PCLK/2^4$ |
| $0101_B$ | $PCLK/2^5$ |
| $0110_B$ | $PCLK/2^6$ |
| $0111_B$ | $PCLK/2^7$ |
| $1000_B$ | $PCLK/2^8$ |
| $1001_B$ | $PCLK/2^9$ |
| $1010_B$ | $PCLK/2^{10}$ |
| $1011_B$ | $PCLK/2^{11}$ |
| $1100_B$ | $PCLK/2^{12}$ |
| $1101_B$ | $PCLK/2^{13}$ |
| $1110_B$ | $PCLK/2^{14}$ |
| $1111_B$ | $PCLK/2^{15}$ |

The above bits are rewritable only when all the counters using CK1 are stopped (TAUDnTE.TAUDnTEm = 0).

**Table 25.15 TAUDnTPS Register Contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 3 to 0 | TAUDnPRS0 [3:0] | Specifies the CK0 clock. |

| TAUDnPRS0[3:0] | CK0 Clock |
|---|---|
| $0000_B$ | PCLK/$2^0$ |
| $0001_B$ | PCLK/$2^1$ |
| $0010_B$ | PCLK/$2^2$ |
| $0011_B$ | PCLK/$2^3$ |
| $0100_B$ | PCLK/$2^4$ |
| $0101_B$ | PCLK/$2^5$ |
| $0110_B$ | PCLK/$2^6$ |
| $0111_B$ | PCLK/$2^7$ |
| $1000_B$ | PCLK/$2^8$ |
| $1001_B$ | PCLK/$2^9$ |
| $1010_B$ | PCLK/$2^{10}$ |
| $1011_B$ | PCLK/$2^{11}$ |
| $1100_B$ | PCLK/$2^{12}$ |
| $1101_B$ | PCLK/$2^{13}$ |
| $1110_B$ | PCLK/$2^{14}$ |
| $1111_B$ | PCLK/$2^{15}$ |

The above bits are rewritable only when all the counters using CK0 are stopped (TAUDnTE.TAUDnTEm = 0).

**NOTE**

The TAUDn clock input PCLK is specified in the first part of this section, **Section 25.1.3, Clock Supply**.

### 25.3.2.2 TAUDnBRS — TAUDn Prescaler Baud Rate Setting Register

This register specifies the division factor of prescaler clock CK3.

CK3 is generated by dividing CK3_PRE by the factor specified in this register plus one. The PCLK prescaler for CK3_PRE is specified in TAUDnTPS.TAUDnPRS3[3:0].

**Access:** This register can be read or written in 8-bit units.

**Address:** <TAUDn_base> + $244_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TAUDnBRS[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.16 TAUDnBRS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | TAUDnBRS[7:0] | Specifies a CK3_PRE clock division factor for generating CK3. |

| TAUDnBRS[7:0] | CK3 Clock |
|---|---|
| $0000\ 0000_B$ | CK3_PRE / 1 |
| $0000\ 0001_B$ | CK3_PRE / 2 |
| $0000\ 0010_B$ | CK3_PRE / 3 |
| $0000\ 0011_B$ | CK3_PRE / 4 |
| ... | ... |
| $1111\ 1110_B$ | CK3_PRE / 255 |
| $1111\ 1111_B$ | CK3_PRE / 256 |

## 25.3.3 Details of TAUDn Control Registers

### 25.3.3.1 TAUDnCDRm — TAUDn Channel Data Register

This register functions either as a compare register or as a capture register, depending on the operating mode specified in TAUDnCMORm.TAUDnMD[4:1].

**Access:** This register can be read or written in 16-bit units.
- When this register functions as a capture register, only reading is possible. Write operation is ignored.
- When this register functions as a compare register, reading and writing is possible.

**Address:** <TAUDn_base> + m × $4_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCDR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.17 TAUDnCDRm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnCDR [15:0] | Data register for capture/compare values |

### 25.3.3.2 TAUDnCNTm — TAUDn Channel Counter Register

This is a channel m counter register.

**Access:** This register is a read-only register that can be read in 16-bit units.

**Address:** <TAUDn_base> + $80_H$ + m × $4_H$

**Value after reset:** $FFFF_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCNT[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.18 TAUDnCNTm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnCNT [15:0] | 16-bit counter value |

A read value depends on a counter value, a changed operating mode, TAUDnTS.TAUDnTSm or TAUDnTT.TAUDnTTm bit value.

The initial read value of the counter depends on an operating mode and how the counter is stopped.

- Stop by a reset
- Stop by a counter stop trigger (TAUDnTT.TAUDnTTm = 1)

**Table 25.19** lists the initial counter read values after the counter is stopped (TAUDnTE.TAUDnTEm = 0) and re-enabled (TAUDnTS.TAUDnTSm = 1).

The table also contains the counter read value one count after the counter is enabled (TAUDnTS.TAUDnTSm = 1) with the counter waiting for a start trigger.

**Table 25.19 TAUDnCNTm Read Values after Re-Enabling Counter**

| Mode Name | Count Method (Up/Down) | TAUDnCNTm Value | | |
|---|---|---|---|---|
| | | Start Value*1 | After Stop Trigger | After One Count |
| Interval timer mode | Count down | $FFFF_H$ | Stop value | — |
| Judge mode | Count down | $FFFF_H$ | Stop value | — |
| Capture mode | Count up | $0000_H$ | Stop value | — |
| Event count mode | Count down | $FFFF_H$ | Stop value | — |
| One-count mode | Count down | $FFFF_H$ | Stop value | Stop value |
| Capture and one-count mode | Count up | $0000_H$ | Stop value | Capture value + 1 (TAUDnCDRm) |
| Judge and one-count mode | Count down | $FFFF_H$ | Stop value | TAUDnCNTm value – 1 |
| Count-up/-down mode | Count down/up | $FFFF_H$ | Stop value | — |
| Pulse one-count mode | Count down | $FFFF_H$ | Stop value | $0000_H$ |
| Count capture mode | Count up | $0000_H$ | Stop value | — |
| Gate count mode | Count down | $FFFF_H$ | Stop value | Stop value |
| Capture and gate count mode | Count up | $0000_H$ | Stop value | Stop value |

Note 1. The value set for TAUDnCNTm when the operating mode is changed after a reset is deasserted.

### 25.3.3.3 TAUDnCMORm — TAUDn Channel Mode OS Register

This register controls channel m operation.

**Access:** This register can be read or written in 16-bit units. Writable only when the counter is stopped (TAUDnTE.TAUDnTEm = 0).

**Address:** <TAUDn_base> + $200_H$ + m × $4_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS [1:0] | | TAUDnCCS [1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS [1:0] | | — | TAUDnMD[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.20 TAUDnCMORm Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Selects an operation clock.<br>An operation clock is used for the TAUDTTINm input edge detection circuit. Setting of TAUDnCMORm.TAUDnCCS[1:0] bits also allows the operation clock to serve as the TAUDnCNTm counter clock. |

| TAUDn CKS1 | TAUDn CKS0 | Selection of Operation Clock |
|---|---|---|
| 0 | 0 | CK0 |
| 0 | 1 | CK1 |
| 1 | 0 | CK2 |
| 1 | 1 | CK3 |

| Bit Position | Bit Name | Function |
|---|---|---|
| 13, 12 | TAUDnCCS[1:0] | Selects a count clock for TAUDnCNTm counter. |

| TAUDn CCS1 | TAUDn CCS0 | Selection of Count Clock |
|---|---|---|
| 0 | 0 | Operation clock specified by TAUDnCMORm.TAUDnCKS[1:0] |
| 0 | 1 | Valid edge of TAUDTTINm input signal |
| 1 | 0 | Setting prohibited |
| 1 | 1 | INTTAUDnIm signal of master channel |

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 | TAUDnMAS | Specifies whether the channel is a master channel or slave channel during synchronous channel operation.<br>0: Slave<br>1: Master<br>This bit setting is valid only for even channels (CHm_even). Odd channels (CHm_odd) are fixed to 0. |

**Table 25.20 TAUDnCMORm Register Contents (2/3)**

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>10 to 8</td><td>TAUDnSTS[2:0]</td><td>Selects an external start trigger.
<table>
<tr><th>TAUDn STS2</th><th>TAUDn STS1</th><th>TAUDn STS0</th><th>Functional Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Software trigger</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Valid edge of TAUDTTINm input signal, which is specified by TAUDnCMURm.TAUDnTIS[1:0].</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Valid edge of TAUDTTINm input signal is used as a start trigger and the opposite edge as a stop trigger.</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Triggers simultaneous rewrite.</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>INTTAUDnIm is the start trigger of master channel</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>INTTAUDnIm of upper channel (m – 1) is the start trigger regardless of master setting</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Dead time output signal of TAUDTTOUTm generating unit</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Up/down output trigger signal of master channel</td></tr>
</table></td></tr>
<tr><td>7, 6</td><td>TAUDnCOS[1:0]</td><td>Specifies the timing for updating capture register TAUDnCDRm and overflow flag TAUDnCSRm.TAUDnOVF of channel m.<br>These bits are valid only when channel m is in capture mode or capture one-count mode.
<table>
<tr><th>TAUDn COS1</th><th>TAUDn COS0</th><th>TAUDnCDRm</th><th>TAUDnCSRm.TAUDnOVF</th></tr>
<tr><td>0</td><td>0</td><td rowspan="2">Updated upon detection of a valid edge of TAUDTTINm input.</td><td>Updated (cleared or set) upon detection of a TAUDTTINm input valid edge:<br>• If a counter overflow has occurred since the last valid edge detection, TAUDnCSRm.TAUDnOVF is set.<br>• If no counter overflow has occurred since the last valid edge detection, TAUDnCSRm.TAUDnOVF is cleared.</td></tr>
<tr><td>0</td><td>1</td><td>Set when a counter overflow occurs, and cleared when TAUDnCSCm.TAUDnCLOV is set to 1.</td></tr>
<tr><td>1</td><td>0</td><td rowspan="2">Updated upon detection of valid edge of TAUDTTINm input and at the occurrence of counter overflow:<br>• Detection of valid edge of TAUDTTINm input: Counter value is written into TAUDnCDRm.<br>• Occurrence of overflow: $FFFF_H$ is loaded into TAUDnCDRm. The next detection of valid edge of TAUDTTINm input is ignored.</td><td>Not set</td></tr>
<tr><td>1</td><td>1</td><td>Set when a counter overflow occurs, and cleared when TAUDnCSCm.TAUDnCLOV is set to 1.</td></tr>
</table></td></tr>
<tr><td>5</td><td>Reserved</td><td>When read, the value after reset is returned. When writing, write the value after reset.</td></tr>
</table>

**Table 25.20 TAUDnCMORm Register Contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 4 to 0 | TAUDnMD[4:0] | Specifies an operating mode. |

| TAUDnMD4 | TAUDnMD3 | TAUDnMD2 | TAUDnMD1 | TAUDnMD0 | Functional Description |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1/0 | Interval timer mode |
| 0 | 0 | 0 | 1 | 1/0 | Judge mode |
| 0 | 0 | 1 | 0 | 1/0 | Capture mode |
| 0 | 0 | 1 | 1 | 0 | Event count mode |
| 0 | 1 | 0 | 0 | 1/0 | One-count mode |
| 0 | 1 | 0 | 1 | 1/0 | Setting prohibited |
| 0 | 1 | 1 | 0 | 0 | Capture and one-count mode |
| 0 | 1 | 1 | 1 | 1/0 | Judge and one-count mode |
| 1 | 0 | 0 | 0 | 0 | Setting prohibited |
| 1 | 0 | 0 | 1 | 0 | Count-up/-down mode |
| 1 | 0 | 1 | 0 | 1/0 | Pulse one-count mode |
| 1 | 0 | 1 | 1 | 1/0 | Count capture mode |
| 1 | 1 | 0 | 0 | 0 | Gate count mode |
| 1 | 1 | 0 | 1 | 0 | Capture and gate count mode |

| Mode | Role of TAUDnMD0 Bit |
|---|---|
| Interval timer mode<br>Capture mode<br>Count capture mode | Specifies whether INTTAUDnIm is generated at the beginning of count operation (when a start trigger is entered) or not.<br>0: INTTAUDnIm is not generated.<br>1: INTTAUDnIm is generated. |
| Event count mode<br>Count-up/-down mode | This bit should be set to 0 (the INTTAUDnIm signal is not output at the beginning of count operation). |
| One-count mode<br>Pulse one-count mode | Enables/disables start trigger detection during counting.<br>0: Disables detection.<br>1: Enables detection.<br>**CAUTION**<br>• INTTAUDnIm signal is not output at the beginning of count operation in one-count mode.<br>• INTTAUDnIm signal is output at the beginning of count operation in pulse one-count mode. |
| Gate count mode | This bit should be set to 0 (disables start trigger detection during counting). |
| Capture and one-count mode<br>Capture and gate count mode | This bit should be set to 0.<br>**CAUTION**<br>INTTAUDnIm signal is not output at the beginning of count operation. In addition, start trigger detected during counting is disabled. |
| Judge mode<br>Judge and one-count mode | Specifies INTTAUDnIm output timing.<br>0: When TAUDnCNTm ≤ TAUDnCDRm<br>1: When TAUDnCNTm > TAUDnCDRm |

### 25.3.3.4 TAUDnCMURm — TAUDn Channel Mode User Register

This register specifies a type of valid edge detection used for TAUDTTINm input.

**Access:** This register can be read or written in 8-bit units.

**Address:** <TAUDn_base> + $C0_H$ + m × $4_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.21 TAUDnCMURm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | Specifies a valid edge of TAUDTTINm input signal. |

| TAUDn TIS1 | TAUDn TIS0 | Functional Description |
|---|---|---|
| 0 | 0 | Falling edge |
| 0 | 1 | Rising edge |
| 1 | 0 | Detection of rising and falling edges (selects low width measurement)<br>Start trigger: Falling edge<br>Stop trigger (capture): Rising edge |
| 1 | 1 | Detection of rising and falling edges (selects high width measurement)<br>Start trigger: Rising edge<br>Stop trigger (capture): Falling edge |

- Edge detection of TAUDTTINm input signal is based on the operation clock selected by TAUDnCMORm.TAUDnCKS[1:0].

### 25.3.3.5 TAUDnCSRm — TAUDn Channel Status Register

This register indicates the count direction and overflow status of channel m counter.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <TAUDn_base> + $140_H$ + m × $4_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnCSF | TAUDnOVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 25.22 TAUDnCSRm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. |
| 1 | TAUDnCSF | Indicates a count direction.<br>0: Count-up<br>1: Count-down<br>The read value of this bit is valid only in the following mode:<br>• Count-up/-down mode |
| 0 | TAUDnOVF | Indicates counter overflow status.<br>0: No overflow occurs.<br>1: Overflow occurs.<br>This bit is used only in the following modes:<br>• Capture mode<br>• Capture and one-count mode<br><br>The function of this bit depends on the setting of control bit TAUDnCMORm.TAUDnCOS[1:0]. |

### 25.3.3.6 TAUDnCSCm — TAUDn Channel Status Clear Register

This is a trigger register for clearing the overflow flag TAUDnCSRm.TAUDnOVF of channel m.

**Access:** This register is a write-only register that can be written in 8-bit units. It is always read as $00_H$.

**Address:** <TAUDn_base> + $180_H$ + m × $4_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | TAUDnCLOV |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 25.23 TAUDnCSCm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | TAUDnCLOV | 0: No function<br>1: Clears overflow flag TAUDnCSRm.TAUDnOVF. |

### 25.3.3.7 TAUDnTS — TAUDn Channel Start Trigger Register

This register enables the counter operation of each channel.

**Access:** This register is a write-only register that can be written in 16-bit units. It is always read as $0000_H$.

**Address:** <TAUDn_base> + $1C4_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnTS15 | TAUDnTS14 | TAUDnTS13 | TAUDnTS12 | TAUDnTS11 | TAUDnTS10 | TAUDnTS09 | TAUDnTS08 | TAUDnTS07 | TAUDnTS06 | TAUDnTS05 | TAUDnTS04 | TAUDnTS03 | TAUDnTS02 | TAUDnTS01 | TAUDnTS00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 25.24 TAUDnTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnTSm | Enables the counter operation of channel m.<br>0: No function<br>1: Enables the counter operation and sets TAUDnTE.TAUDnTEm to 1.<br>The counter operation is only enabled when TAUDnTE.TAUDnTEm is set to 1.<br>Whether counting is started or not depends on a selected operating mode. |

### 25.3.3.8 TAUDnTE — TAUDn Channel Enable Status Register

This register enables/disables a counter operation.

**Access:** This register is a read-only register that can be read in 16-bit units.

**Address:** <TAUDn_base> + $1C0_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnTE15 | TAUDnTE14 | TAUDnTE13 | TAUDnTE12 | TAUDnTE11 | TAUDnTE10 | TAUDnTE09 | TAUDnTE08 | TAUDnTE07 | TAUDnTE06 | TAUDnTE05 | TAUDnTE04 | TAUDnTE03 | TAUDnTE02 | TAUDnTE01 | TAUDnTE00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.25 TAUDnTE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnTEm | Enables/disables the counter operation of channel m.<br>0: Disables counter operation.<br>1: Enables counter operation.<br>This bit is set to 1 when trigger input of TAUDnTSSTm (synchronous channel start trigger signal) is detected or when TAUDnTS.TAUDnTSm is set to 1. This bit is set to 0 when TAUDnTT.TAUDnTTm is set to 1. |

### 25.3.3.9 TAUDnTT — TAUDn Channel Stop Trigger Register

This register stops the counter operation of each channel.

**Access:** This register is a write-only register that can be written in 16-bit units. It is always read as $0000_H$.

**Address:** <TAUDn_base> + $1C8_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnTT15 | TAUDnTT14 | TAUDnTT13 | TAUDnTT12 | TAUDnTT11 | TAUDnTT10 | TAUDnTT09 | TAUDnTT08 | TAUDnTT07 | TAUDnTT06 | TAUDnTT05 | TAUDnTT04 | TAUDnTT03 | TAUDnTT02 | TAUDnTT01 | TAUDnTT00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 25.26 TAUDnTT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnTTm | Stops the counter operation of channel m.<br>0: No function<br>1: Stops the counter operation and resets TAUDnTE.TAUDnTEm.<br>TAUDnCNTm, TAUDnTO.TAUDnTOm, and TAUDTTOUTm retain the values provided before the counter is stopped. |

## 25.3.4 Details of TAUDn Simultaneous Rewrite Registers

### 25.3.4.1 TAUDnRDE — TAUDn Channel Reload Data Enable Register

This register enables/disables simultaneous rewrite of TAUDnCDRm/TAUDnTOLm data register.

**Access:** This register can be read or written in 16-bit units. Writable only while TAUDnTE.TAUDnTEm = 0.

**Address:** <TAUDn_base> + $260_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn RDE15 | TAUDn RDE14 | TAUDn RDE13 | TAUDn RDE12 | TAUDn RDE11 | TAUDn RDE10 | TAUDn RDE09 | TAUDn RDE08 | TAUDn RDE07 | TAUDn RDE06 | TAUDn RDE05 | TAUDn RDE04 | TAUDn RDE03 | TAUDn RDE02 | TAUDn RDE01 | TAUDn RDE00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.27 TAUDnRDE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnRDEm | Enables/disables simultaneous rewrite of the data register of channel m.<br>0: Disables simultaneous rewrite<br>1: Enables simultaneous rewrite |

### 25.3.4.2 TAUDnRDS — TAUDn Channel Reload Data Control Channel Select Register

This register selects a channel that controls simultaneous rewrite.

**Access:** This register can be read or written in 16-bit units. Writable only while TAUDnTE.TAUDnTEm = 0.

**Address:** <TAUDn_base> + $268_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn RDS15 | TAUDn RDS14 | TAUDn RDS13 | TAUDn RDS12 | TAUDn RDS11 | TAUDn RDS10 | TAUDn RDS09 | TAUDn RDS08 | TAUDn RDS07 | TAUDn RDS06 | TAUDn RDS05 | TAUDn RDS04 | TAUDn RDS03 | TAUDn RDS02 | TAUDn RDS01 | TAUDn RDS00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.28 TAUDnRDS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnRDSm | Selects a channel that controls a simultaneous rewrite trigger.<br>0: Master channel<br>1: Another upper channel |

### 25.3.4.3 TAUDnRDM — TAUDn Channel Reload Data Mode Register

This register selects the timing for generating a simultaneous rewrite control signal.

**Access:** This register can be read or written in 16-bit units. Writable only while TAUDnTE.TAUDnTEm = 0.

**Address:** <TAUDn_base> + $264_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn RDM15 | TAUDn RDM14 | TAUDn RDM13 | TAUDn RDM12 | TAUDn RDM11 | TAUDn RDM10 | TAUDn RDM09 | TAUDn RDM08 | TAUDn RDM07 | TAUDn RDM06 | TAUDn RDM05 | TAUDn RDM04 | TAUDn RDM03 | TAUDn RDM02 | TAUDn RDM01 | TAUDn RDM00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.29 TAUDnRDM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnRDMm | Selects the timing for generating a simultaneous rewrite trigger signal.<br>0: When the master channel counter starts to count<br>1: At the peak of cycle of triangular wave<br>These bit settings are applied only when TAUDnRDE.TAUDnRDEm = 1 and TAUDnRDS.TAUDnRDSm = 0. |

### 25.3.4.4 TAUDnRDC — TAUDn Channel Reload Data Control Register

This register specifies a channel which generates an INTTAUDnIm signal to trigger simultaneous rewrite.

**Access:** This register can be read or written in 16-bit units. Writable only while TAUDnTE.TAUDnTEm = 0.

**Address:** <TAUDn_base> + $26C_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn RDC15 | TAUDn RDC14 | TAUDn RDC13 | TAUDn RDC12 | TAUDn RDC11 | TAUDn RDC10 | TAUDn RDC09 | TAUDn RDC08 | TAUDn RDC07 | TAUDn RDC06 | TAUDn RDC05 | TAUDn RDC04 | TAUDn RDC03 | TAUDn RDC02 | TAUDn RDC01 | TAUDn RDC00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.30 TAUDnRDC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnRDCm | Specifies whether the channel generates a simultaneous rewrite trigger signal or not.<br>0: Does not operate as a simultaneous rewrite trigger channel.<br>1: Operates as a simultaneous rewrite trigger channel.<br>These bit settings are applied only when TAUDnRDE.TAUDnRDEm = 1 and TAUDnRDS.TAUDnRDSm = 1. |

### 25.3.4.5 TAUDnRDT — TAUDn Channel Reload Data Trigger Register

This register triggers a simultaneous rewrite enabling state.

**Access:** This register is a write-only register that can be written in 16-bit units. It is always read as $0000_H$.

**Address:** <TAUDn_base> + $044_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn RDT15 | TAUDn RDT14 | TAUDn RDT13 | TAUDn RDT12 | TAUDn RDT11 | TAUDn RDT10 | TAUDn RDT09 | TAUDn RDT08 | TAUDn RDT07 | TAUDn RDT06 | TAUDn RDT05 | TAUDn RDT04 | TAUDn RDT03 | TAUDn RDT02 | TAUDn RDT01 | TAUDn RDT00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 25.31 TAUDnRDT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnRDTm | Triggers a simultaneous rewrite enabling state.<br>0: No function<br>1: The simultaneous rewrite enabling flag (TAUDnRSFm) is set to 1. The system waits for a simultaneous rewrite trigger.<br>These bits only apply when:<br>• TAUDnRDE.TAUDnRDEm = 1 |

### 25.3.4.6 TAUDnRSF — TAUDn Channel Reload Status Register

This flag register indicates simultaneous rewrite status.

**Access:** This register is a read-only register that can be read in 16-bit units.

**Address:** <TAUDn_base> + $048_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn RSF15 | TAUDn RSF14 | TAUDn RSF13 | TAUDn RSF12 | TAUDn RSF11 | TAUDn RSF10 | TAUDn RSF09 | TAUDn RSF08 | TAUDn RSF07 | TAUDn RSF06 | TAUDn RSF05 | TAUDn RSF04 | TAUDn RSF03 | TAUDn RSF02 | TAUDn RSF01 | TAUDn RSF00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.32 TAUDnRSF Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnRSFm | Indicates simultaneous rewrite status.<br>0: Indicates that simultaneous rewrite has been completed due to the generation of simultaneous rewrite trigger.<br>1: Indicates that the system waits for a simultaneous rewrite trigger in the simultaneous rewrite enabling state (TAUDnRDTm = 1). |

## 25.3.5 Details of TAUDn Output Registers

### 25.3.5.1 TAUDnTOE — TAUDn Channel Output Enable Register

This register enables/disables the independent channel output mode controlled by software.

**Access:** This register can be read or written in 16-bit units.

**Address:** <TAUDn_base> + $5C_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn TOE15 | TAUDn TOE14 | TAUDn TOE13 | TAUDn TOE12 | TAUDn TOE11 | TAUDn TOE10 | TAUDn TOE09 | TAUDn TOE08 | TAUDn TOE07 | TAUDn TOE06 | TAUDn TOE05 | TAUDn TOE04 | TAUDn TOE03 | TAUDn TOE02 | TAUDn TOE01 | TAUDn TOE00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.33 TAUDnTOE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnTOEm | Enables/disables the independent channel output function.<br>0: Disables the independent timer output function (controlled by software).<br>1: Enables the independent timer output function. |

### 25.3.5.2 TAUDnTO — TAUDn Channel Output Register

This register specifies and reads a TAUDTTOUTm level.

**Access:** This register can be read or written in 16-bit units.

**Address:** <TAUDn_base> + $58_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn TO15 | TAUDn TO14 | TAUDn TO13 | TAUDn TO12 | TAUDn TO11 | TAUDn TO10 | TAUDn TO09 | TAUDn TO08 | TAUDn TO07 | TAUDn TO06 | TAUDn TO05 | TAUDn TO04 | TAUDn TO03 | TAUDn TO02 | TAUDn TO01 | TAUDn TO00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.34 TAUDnTO Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnTOm | Specifies and reads a TAUDTTOUTm level.<br>0: Low level<br>1: High level<br>Only TAUDnTOm bits for which Independent Channel Output function is disabled (TAUDnTOEm = 0) can be written. |

### 25.3.5.3 TAUDnTOM — TAUDn Channel Output Mode Register

This register specifies the output mode of each channel.

**Access:** This register can be read or written in 16-bit units. Writable only while the counter is stopped (TAUDnTE.TAUDnTEm = 0).

**Address:** <TAUDn_base> + $248_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn TOM15 | TAUDn TOM14 | TAUDn TOM13 | TAUDn TOM12 | TAUDn TOM11 | TAUDn TOM10 | TAUDn TOM09 | TAUDn TOM08 | TAUDn TOM07 | TAUDn TOM06 | TAUDn TOM05 | TAUDn TOM04 | TAUDn TOM03 | TAUDn TOM02 | TAUDn TOM01 | TAUDn TOM00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.35 TAUDnTOM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnTOMm | Specifies an output mode.<br>0: Independent channel operation<br>1: Synchronous channel operation |

### 25.3.5.4 TAUDnTOC — TAUDn Channel Output Configuration Register

This register specifies the output mode of each channel in combination with TAUDnTOMm.

**Access:** This register can be read or written in 16-bit units. Writable only while the counter is stopped (TAUDnTE.TAUDnTEm = 0).

**Address:** <TAUDn_base> + $24C_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn TOC15 | TAUDn TOC14 | TAUDn TOC13 | TAUDn TOC12 | TAUDn TOC11 | TAUDn TOC10 | TAUDn TOC09 | TAUDn TOC08 | TAUDn TOC07 | TAUDn TOC06 | TAUDn TOC05 | TAUDn TOC04 | TAUDn TOC03 | TAUDn TOC02 | TAUDn TOC01 | TAUDn TOC00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.36 TAUDnTOC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnTOCm | Specifies an output mode.<br>0: Operating mode 1<br>1: Operating mode 2<br>As listed below, the output mode depends on the setting of TAUDnTOM.TAUDnTOMm. |

| TAUDn TOMm | TAUDn TOCm | Functional Description |
|---|---|---|
| 0 | 0 | Toggle mode: Toggle operation is conducted when INTTAUDnIm occurs. |
| 0 | 1 | Set/reset mode: Set when INTTAUDnIm occurs at the beginning of count operation, and reset when INTTAUDnIm is caused by detection of a match between TAUDnCNTm and TAUDnCDRm. |
| 1 | 0 | Synchronous channel operating mode 1: Set when INT occurs on master channels, and reset when INT occurs on slave channels. |
| 1 | 1 | Synchronous channel operating mode 2: Set when INTTAUDnIm occurs in count-down status, and reset when INTTAUDnIm occurs in count-up status. |

### 25.3.5.5 TAUDnTOL — TAUDn Channel Output Active Level Register

This register specifies the output logic of channel output bit (TAUDnTO.TAUDnTOm).

**Access:** This register can be read or written in 16-bit units.

**Address:** <TAUDn_base> + $040_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn TOL15 | TAUDn TOL14 | TAUDn TOL13 | TAUDn TOL12 | TAUDn TOL11 | TAUDn TOL10 | TAUDn TOL09 | TAUDn TOL08 | TAUDn TOL07 | TAUDn TOL06 | TAUDn TOL05 | TAUDn TOL04 | TAUDn TOL03 | TAUDn TOL02 | TAUDn TOL01 | TAUDn TOL00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.37 TAUDnTOL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnTOLm | Specifies the output logic of channel m output bit (TAUDnTO.TAUDnTOm).<br>0: Positive logic (active high)<br>1: Negative logic (active low)<br>The setting of these bits applies to all channel output modes other than independent channel output mode controlled by software. |

## 25.3.6 Details of TAUDn Dead Time Output Registers

### 25.3.6.1 TAUDnTDE — TAUDn Channel Dead Time Output Enable Register

This register enables/disables the dead time operation of every channel.

**Access:** This register can be read or written in 16-bit units. Writable only while the counter is stopped (TAUDnTE.TAUDnTEm = 0).

**Address:** <TAUDn_base> + $250_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn TDE15 | TAUDn TDE14 | TAUDn TDE13 | TAUDn TDE12 | TAUDn TDE11 | TAUDn TDE10 | TAUDn TDE09 | TAUDn TDE08 | TAUDn TDE07 | TAUDn TDE06 | TAUDn TDE05 | TAUDn TDE04 | TAUDn TDE03 | TAUDn TDE02 | TAUDn TDE01 | TAUDn TDE00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.38 TAUDnTDE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnTDEm | Enables/disables the dead time control operation of channel m.<br>0: Disables dead time operation<br>1: Enables dead time operation.<br>The same setting should be made for both even and odd slave channels in pairs.<br>These bit settings are applied when:<br>• TAUDnTOE.TAUDnTOEm, TAUDnTOM.TAUDnTOMm, TAUDnTOC.TAUDnTOCm = 1 |

### 25.3.6.2 TAUDnTDM — TAUDn Channel Dead Time Output Mode Register

This register specifies the timing to add dead time during dead time output.

**Access:** This register can be read or written in 16-bit units. Writable only while the counter is stopped (TAUDnTE.TAUDnTEm = 0).

**Address:** <TAUDn_base> + $254_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn TDM15 | TAUDn TDM14 | TAUDn TDM13 | TAUDn TDM12 | TAUDn TDM11 | TAUDn TDM10 | TAUDn TDM09 | TAUDn TDM08 | TAUDn TDM07 | TAUDn TDM06 | TAUDn TDM05 | TAUDn TDM04 | TAUDn TDM03 | TAUDn TDM02 | TAUDn TDM01 | TAUDn TDM00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.39 TAUDnTDM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnTDMm | Specifies the timing to add dead time during dead time output.<br>0: When detecting the duty cycle of an upper even channel (duty dead time output).<br>1: When detecting the TIN input edge of a lower odd channel (one-phase dead time output).<br>The same setting should be made for both even and odd slave channels in pairs.<br>These bit settings are applied when:<br>• TAUDnTOE.TAUDnTOEm, TAUDnTOM.TAUDnTOMm, TAUDnTOC.TAUDnTOCm, TAUDnTDE.TAUDnTDEm = 1 |

### 25.3.6.3 TAUDnTDL — TAUDn Channel Dead Time Output Level Register

This register selects a phase in which dead time is added.

**Access:** This register can be read or written in 16-bit units.

**Address:** <TAUDn_base> + $54_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn TDL15 | TAUDn TDL14 | TAUDn TDL13 | TAUDn TDL12 | TAUDn TDL11 | TAUDn TDL10 | TAUDn TDL09 | TAUDn TDL08 | TAUDn TDL07 | TAUDn TDL06 | TAUDn TDL05 | TAUDn TDL04 | TAUDn TDL03 | TAUDn TDL02 | TAUDn TDL01 | TAUDn TDL00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.40 TAUDnTDL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnTDLm | Selects a phase in which dead time is added.<br>0: Normal phase<br>1: Reverse phase<br>These bit settings are applied when:<br>• TAUDnTOE.TAUDnTOEm, TAUDnTOM.TAUDnTOMm, TAUDnTOC.TAUDnTOCm, TAUDnTDE.TAUDnTDEm = 1 |

## 25.3.7 Details of TAUDn Real-time/Modulation Output Registers

### 25.3.7.1 TAUDnTRE — TAUDn Channel Real-time Output Enable Register

This register enables/disables real-time output.

**Access:** This register can be read or written in 16-bit units. Writable only while TAUDnTE.TAUDnTEm = 0.

**Address:** <TAUDn_base> + $258_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn TRE15 | TAUDn TRE14 | TAUDn TRE13 | TAUDn TRE12 | TAUDn TRE11 | TAUDn TRE10 | TAUDn TRE09 | TAUDn TRE08 | TAUDn TRE07 | TAUDn TRE06 | TAUDn TRE05 | TAUDn TRE04 | TAUDn TRE03 | TAUDn TRE02 | TAUDn TRE01 | TAUDn TRE00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.41 TAUDnTRE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnTREm | Enables or disables real-time output of channel m.<br>0: Disables real-time output<br>1: Enables real-time output.<br>These bit settings are applied only when TAUDnTOE.TAUDnTOEm = 1. When TAUDnTRE.TAUDnTREm = 0, TAUDTTOUTm is not affected by real-time output.<br>When TAUDnTRE.TAUDnTREm = 1, TAUDTTOUTm outputs the value of real-time output bit TAUDnTRO.TAUDnTROm in response to a timer operation. |

### 25.3.7.2 TAUDnTRC — TAUDn Channel Real-time Output Control Register

This register controls the real-time output trigger of each channel.

**Access:** This register can be read or written in 16-bit units. Writable only while TAUDnTE.TAUDnTEm = 0.

**Address:** <TAUDn_base> + $25C_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn TRC15 | TAUDn TRC14 | TAUDn TRC13 | TAUDn TRC12 | TAUDn TRC11 | TAUDn TRC10 | TAUDn TRC09 | TAUDn TRC08 | TAUDn TRC07 | TAUDn TRC06 | TAUDn TRC05 | TAUDn TRC04 | TAUDn TRC03 | TAUDn TRC02 | TAUDn TRC01 | TAUDn TRC00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.42 TAUDnTRC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnTRCm | Specifies a channel on which the real-time output trigger for channel m is generated.<br>0: Next upper channel with this bit set to 1<br>1: Channel m<br>These bit settings are applied only when TAUDnTRE.TAUDnTREm = 1. |

### 25.3.7.3 TAUDnTRO — TAUDn Channel Real-time Output Register

This register sets a value which is output to TAUDTTOUTm.

**Access:** This register can be read or written in 16-bit units.

**Address:** <TAUDn_base> + $04C_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn TRO15 | TAUDn TRO14 | TAUDn TRO13 | TAUDn TRO12 | TAUDn TRO11 | TAUDn TRO10 | TAUDn TRO09 | TAUDn TRO08 | TAUDn TRO07 | TAUDn TRO06 | TAUDn TRO05 | TAUDn TRO04 | TAUDn TRO03 | TAUDn TRO02 | TAUDn TRO01 | TAUDn TRO00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.43 TAUDnTRO Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnTROm | Sets a value which is output to TAUDTTOUTm.<br>0: Low<br>1: High<br>TAUDnTROm value is not output to TAUDTTOUTm when TAUDnTRE.TAUDnTREm = 0, even if a real-time output trigger occurs. |

### 25.3.7.4 TAUDnTME — TAUDn Channel Modulation Output Enable Register

This register enables/disables modulation output for timer output and real-time output.

**Access:** This register can be read or written in 16-bit units.

**Address:** <TAUDn_base> + $050_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDn TME15 | TAUDn TME14 | TAUDn TME13 | TAUDn TME12 | TAUDn TME11 | TAUDn TME10 | TAUDn TME09 | TAUDn TME08 | TAUDn TME07 | TAUDn TME06 | TAUDn TME05 | TAUDn TME04 | TAUDn TME03 | TAUDn TME02 | TAUDn TME01 | TAUDn TME00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.44 TAUDnTME Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TAUDnTMEm | Enables/disables modulation output for timer output and real-time output of channel m.<br>0: Disables modulation<br>1: Enables modulation<br>These bit settings are applied only when TAUDnTOE.TAUDnTOEm and TAUDnTRE.TAUDnTREm = 1. |

## 25.3.8 TAUDn Emulation Register

### 25.3.8.1 TAUDnEMU — TAUDn Emulation Register

This register controls SVSTOP operations.

**Access:** This register can be read or written in 8-bit units.
Perform write operations when the counter is stopped (TAUDnTE.TAUDnTEm = 0) and (EPC.SVSTOP = 0).

**Address:** <TAUDn_base> + $290_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TAUDnSVSDIS | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R |

**Table 25.45 TAUDnEMU Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | TAUDnSVSDIS | When EPC.SVSTOP bit = 0:<br>Supply of the count clock continues when the debugger takes control of the microcontroller (as in the breakpoint), regardless of the value of this bit (1 or 0).<br><br>When EPC.SVSTOP bit = 1:<br>0: The count clock is stopped when the debugger takes control of the microcontroller (as in the breakpoint).<br>1: Supply of the count clock continues when the debugger takes control of the microcontroller (as in the breakpoint). |
| 6 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

## 25.4 Operating Procedure

The following lists the general operation procedure for the TAUDn.

After reset release, the operation of each channel is stopped. Clock supply is started and writing to each register is enabled. All circuits and registers of all channels are initialized. The control register of TAUDTTOUTm is also initialized and outputs a low level.

(1) Set the TAUDnTPS and TAUDnBRS registers to specify the clock frequency of CK0 to CK3.

(2) Configure the desired TAUDn function:
- Set the operation mode
- Set the channel output mode
- Set any other control bits

(3) Enable the counter by setting the TAUDnTS.TAUDnTSm bit to 1.
The counter starts to count immediately, or when an appropriate trigger is detected, depending on the bit settings.

(4) If desired, and if possible for the configured function, stop the counter or perform a forced restart operation during count operation. The counter can be stopped by setting the TAUDnTT.TAUDnTTm bit to 1. The counter can be forcibly restarted by setting the TAUDnTS.TAUDnTSm bit to 1.

(5) Stop the function by setting the TAUDnTT.TAUDnTTm bit to 1.

**NOTES**

1. A detailed description of the required control bits and the operation of the individual functions are given in **Section 25.12, Independent Channel Operation Functions** and **Section 25.15, Synchronous Channel Operation Functions**.
2. The function can be changed while the counter is stopped (TAUDnTE.TAUDnTEm = 0).

## 25.5 Concepts of Synchronous Channel Operation

The synchronous channel operation function is implemented using a combination of channel groups (consisted of master and slave channels). Several rules apply to the settings of channels. These rules are detailed in **Section 25.5.1, Rules of Synchronous Channel Operation**.

Two special features for synchronous channel operation are detailed in the following:

- **Section 25.5.2, Simultaneous Start and Stop of Synchronous Channel Counters**
- **Section 25.6, Simultaneous Rewrite**

### 25.5.1 Rules of Synchronous Channel Operation

#### Number of master and slave channels

- Only even channels (CH0, CH2, CH4, …) can be set as master channels. Any channel apart from CH0 can be set as a slave channel.
- Only channels lower than the master channel can be set as slave channels, and multiple slave channels can be set for one master channel.
  Example: If CH2 is a master channel, CH3 and the lower channels (CH3, CH4, CH5, …) can be set as slave channels.
- If multiple master channels are used, slave channels cannot cross the master channels.
  Example: If CH0 and CH4 are master channels, CH1 to CH3 can be set as slave channels for CH0, but CH5 to CH15 cannot.

#### Operation clock

- The same operation clock must be set for the master channel and the synchronized slave channel. This is achieved by setting the TAUDnCMORm.TAUDnCKS[1:0] bits of the slave and master channel.

The basic concepts of master/slave usage and operation clocks are illustrated in **Figure 25.4**.

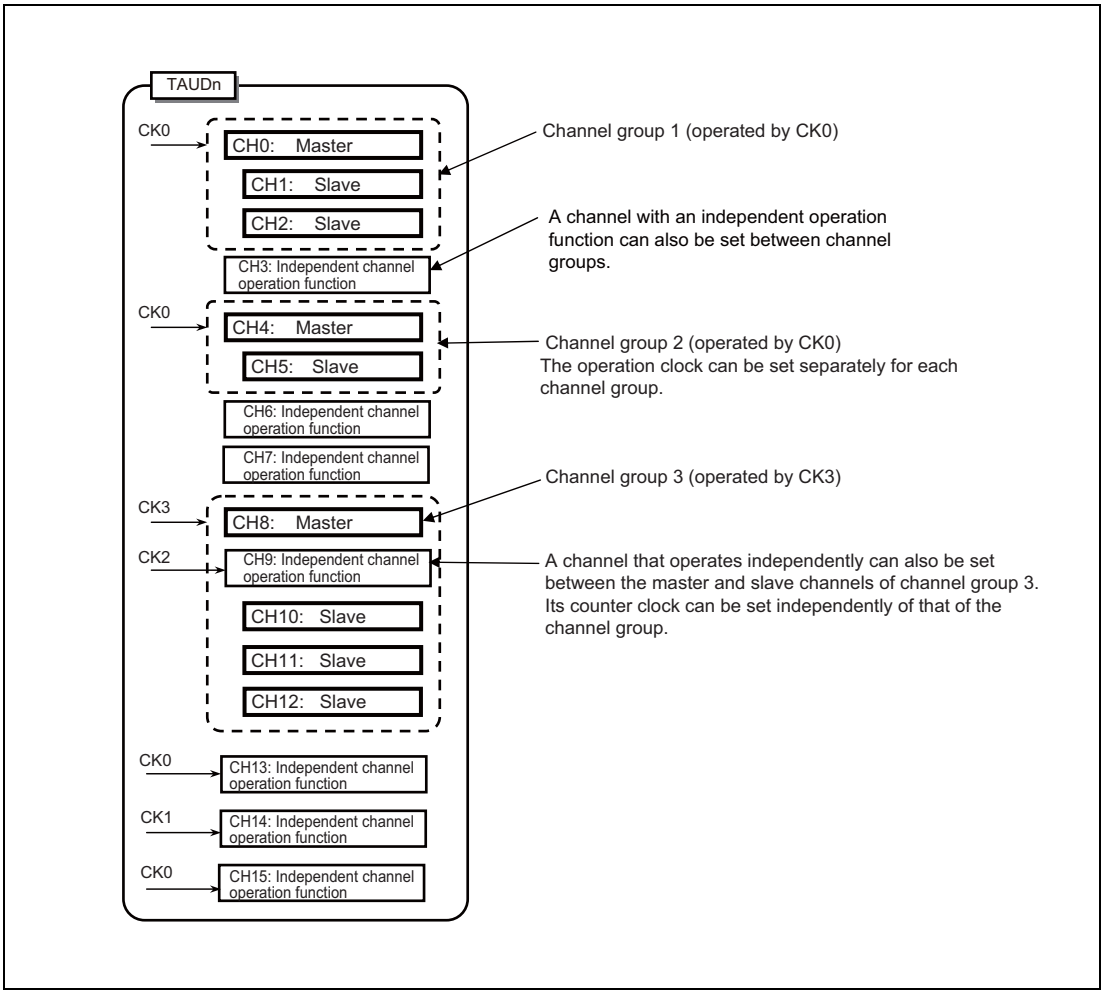



**Figure 25.4 Grouping of Channels and Assignment of Count Clocks**

### Control trigger signal for master/slave channels

- Master channels can output control trigger signals to slave channels.
- Slave channels can use control trigger signals from master channels but cannot output control trigger signals for their own to lower channels.
- Master channels cannot use control trigger signals from upper master channels.

### 25.5.2 Simultaneous Start and Stop of Synchronous Channel Counters

Channels that are operated synchronously can be started and stopped simultaneously within the same unit and between the units.

#### 25.5.2.1 Simultaneous Start and Stop within the Same Unit

- To simultaneously start synchronized channels, the TAUDnTS.TAUDnTSm bits of the channels should be set at the same time.
- To simultaneously stop synchronized channels, the TAUDnTT.TAUDnTTm bits of the channels should be set at the same time.

Setting to the TAUDnTS.TAUDnTSm bits to 1 also sets the corresponding TAUDnTE.TAUDnTEm bits to 1, enabling counting. The count start timing depends on operating mode.

#### 25.5.2.2 Simultaneous Start between the Units

Counters in different units can also be started simultaneously if the corresponding counters are enabled before receiving the simultaneous trigger signal.

For details about how to perform simultaneous start between the units, see **Section 29.8, Simultaneous Start Trigger Function**.

# 25.6 Simultaneous Rewrite

## 25.6.1 Overview of Operations

Simultaneous rewrite describes the ability to change the compare/start value and the output logic of multiple channels at the same time.

The corresponding data and control registers (TAUDnCDRm and TAUDnTOLm) can nevertheless be written at any time. The new value does not affect the counter operation or the output signal until simultaneous rewrite is triggered.

Simultaneous rewrite can be triggered by:

- The counter on the master channel or upper channel (depending on the selected operation mode) reaching a certain value
- INTTAUDnIm being issued on the upper channel specified by TAUDnRDC.TAUDnRDCm

There are four methods for simultaneous rewrite. These are listed in **Table 25.46**, along with how to specify them and when they cause simultaneous rewrite to be triggered.

**Table 25.46 Simultaneous Rewrite Methods and when They are Triggered**

| Method | Simultaneous Rewrite Triggered when | TAUDnRDE. TAUDnRDEm | TAUDnRDS. TAUDnRDSm | TAUDnRDM. TAUDnRDMm |
|---|---|---|---|---|
| — | No simultaneous rewrite | 0 | 0 | 0 |
| A | The master channel (re)starts counting | 1 | 0 | 0 |
| B | Counting is started in the master channel. The master channel starts counting down at the peak of triangular cycle of the corresponding slave channel. | 1 | 0 | 1 |
| C1 | INTTAUDnIm is generated on an upper channel specified by TAUDnRDC.TAUDnRDCm | 1 | 1 | 0/1 |
| C2 | INTTAUDnIm is generated on an upper channel specified by TAUDnRDC.TAUDnRDCm that in turn is triggered by an external signal | 1 | 1 | 0/1 |

**Table 25.47** lists which of these four methods is available for each channel operation function. For more information about the individual channel operation functions, see the corresponding sections in **Section 25.14, Independent Channel Simultaneous Rewrite Functions**, **Section 25.15, Synchronous Channel Operation Functions**, and **Section 25.16, Synchronous Non-Complementary and Complementary Modulation Output Functions**.

**Table 25.47 Channel Functions and the Methods They Use for Simultaneous Rewrite**

| Function | A | B | C1 | C2 | TAUDnTOL. TAUDnTOLm |
|---|---|---|---|---|---|
| Simultaneous Rewrite Trigger Output Function Type 1 | | | √ | | |
| PWM Output Function | √ | | √ | | √ |
| One-Shot Pulse Output Function | √ | | | | |
| Trigger Start PWM Output Function | √ | | | √ | |
| Delay Pulse Output Function | √ | | | | |
| Triangle PWM Output Function | | √ | √ | | √ |
| Triangle PWM Output Function with Dead Time | | √ | √ | | |
| Interrupt Request Signals Culling Function | √ | √ | √ | | |
| AD Conversion Trigger Output Function Type 1 | √ | | √ | | |
| AD Conversion Trigger Output Function Type 2 | | √ | √ | | |
| Non-Complementary Modulation Output Function Type 1 | √ | | √ | | |
| Non-Complementary Modulation Output Function Type 2 | | √ | √ | | |
| Complementary Modulation Output Function | | √ | √ | | |

**Note:** √: Available, (Blank): Unavailable

### 25.6.2 How to Control Simultaneous Rewrite

**Figure 25.5** shows the general procedure for simultaneous rewrite. The three main blocks (initial settings, start and counter count operation, and simultaneous rewrite) are explained afterwards.

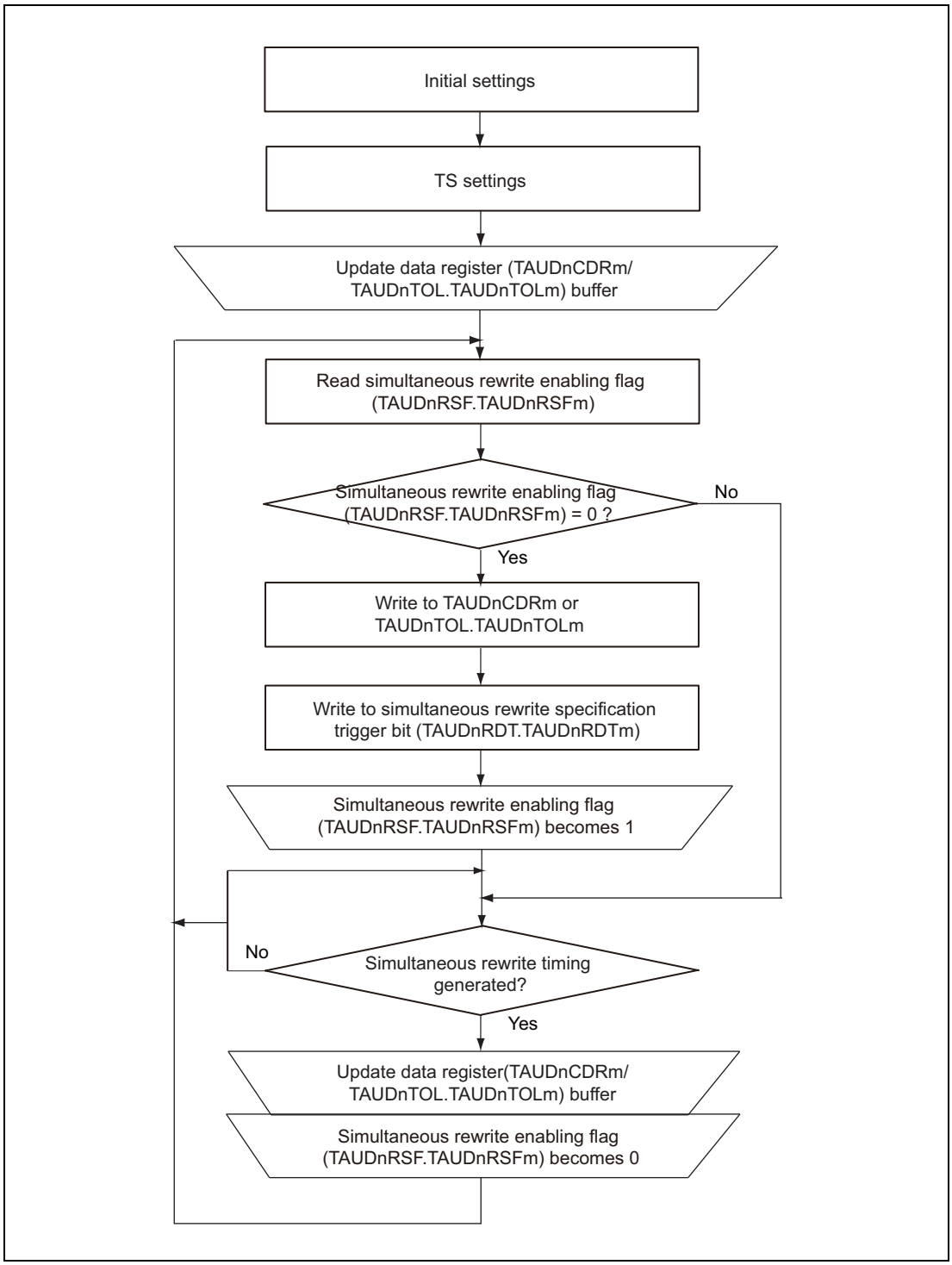



**Figure 25.5 General Procedure for Simultaneous Rewrite**

#### 25.6.2.1 Initial Settings

- To enable simultaneous rewrite in channel m, set TAUDnRDE.TAUDnRDEm = 1
- To select the type of simultaneous rewrite, set TAUDnRDM.TAUDnRDMm and TAUDnRDS.TAUDnRDSm according to the values listed in **Table 25.46, Simultaneous Rewrite Methods and when They are Triggered**.
- Specify a simultaneous rewrite trigger channel by using TAUDnRDC.TAUDnRDCm. (Prerequisite: TAUDnRDS.TAUDnRDSm has been set to the upper channel.)

#### 25.6.2.2 Start Counter and Count Operation

- To start all the TAUDnCNTm counters of the channel group, set the corresponding TAUDnTS.TAUDnTSm bits to 1. The values of TAUDnTOL.TAUDnTOLm and the data registers (TAUDnCDRm) are loaded into the corresponding TAUDnTOL.TAUDnTOLm buffer (TAUDnTOL.TAUDnTOLm buf) and data buffer registers (TAUDnCDRm buf) and the counters start.
- Setting the reload data trigger bit (TAUDnRDT.TAUDnRDTm) to 1 sets the reload flag (TAUDnRSF.TAUDnRSFm) to 1, enabling simultaneous rewrite. TAUDnRSF.TAUDnRSFm remains set to 1 until simultaneous rewrite is completed.
- When the specified trigger for simultaneous rewrite is detected, the TAUDnRSF.TAUDnRSFm bit is checked to see if simultaneous rewrite is enabled (TAUDnRSF.TAUDnRSFm = 1). If it is, simultaneous rewrite is carried out. Otherwise the simultaneous rewrite is not carried out and waits for the next trigger detection.

#### 25.6.2.3 Simultaneous Rewrite

- When simultaneous rewrite is enabled (TAUDnRSF.TAUDnRSFm = 1) and the simultaneous rewrite trigger is detected, the current values of the data registers are copied to their buffers. These values are then loaded into the corresponding counters and are applied the next time the counter starts or restarts.
- When simultaneous rewrite is complete, the TAUDnRSF.TAUDnRSFm bit is set to 0, and the system awaits the next simultaneous rewrite trigger.

### 25.6.3 Other General Rules of Simultaneous Rewrite

The following rules also apply:

- TAUDnRDE.TAUDnRDEm, TAUDnRDS.TAUDnRDSm, TAUDnRDM.TAUDnRDMm, and TAUDnRDC.TAUDnRDCm cannot be changed while the counter is in operation (TAUDnTE.TAUDnTEm = 1).
- TAUDnTOL.TAUDnTOLm can only be rewritten during operation with PWM output function or triangle PWM output function. For all other output functions, TAUDnTOL.TAUDnTOLm should be written before the counter starts. If it is rewritten while any other function is used, TAUDTTOUTm outputs an invalid wave.
- When an upper channel is used as a channel issuing the simultaneous rewrite trigger (TAUDnRDS.TAUDnRDSm = 1), the TAUDnRDC.TAUDnRDCm bit controls all the lower channels. This means that if the TAUDnRDC.TAUDnRDCm bits of CH2 and CH7 are set to 1 and the TAUDnRDC.TAUDnRDCm bits of other channels are set to 0, CH2 and CH7 serve as simultaneous rewrite trigger generation channels. CH2 controls the lower channels CH3 to CH6, and CH7 controls the lower channels CH8 to CH15.
- If simultaneous rewrite is enabled and an upper channel is selected for the simultaneous rewrite trigger (TAUDnRDE.TAUDnRDEm and TAUDnRDS.TAUDnRDSm = 1) but no upper channel is set (TAUDnRDC.TAUDnRDC[15:0] = 0), simultaneous rewrite cannot take place.

### 25.6.4 Types of Simultaneous Rewrite

In the following section, the four simultaneous rewrite methods are explained using timing diagrams.

#### 25.6.4.1 Simultaneous Rewrite when the Master Channel (Re)starts Counting (Method A)

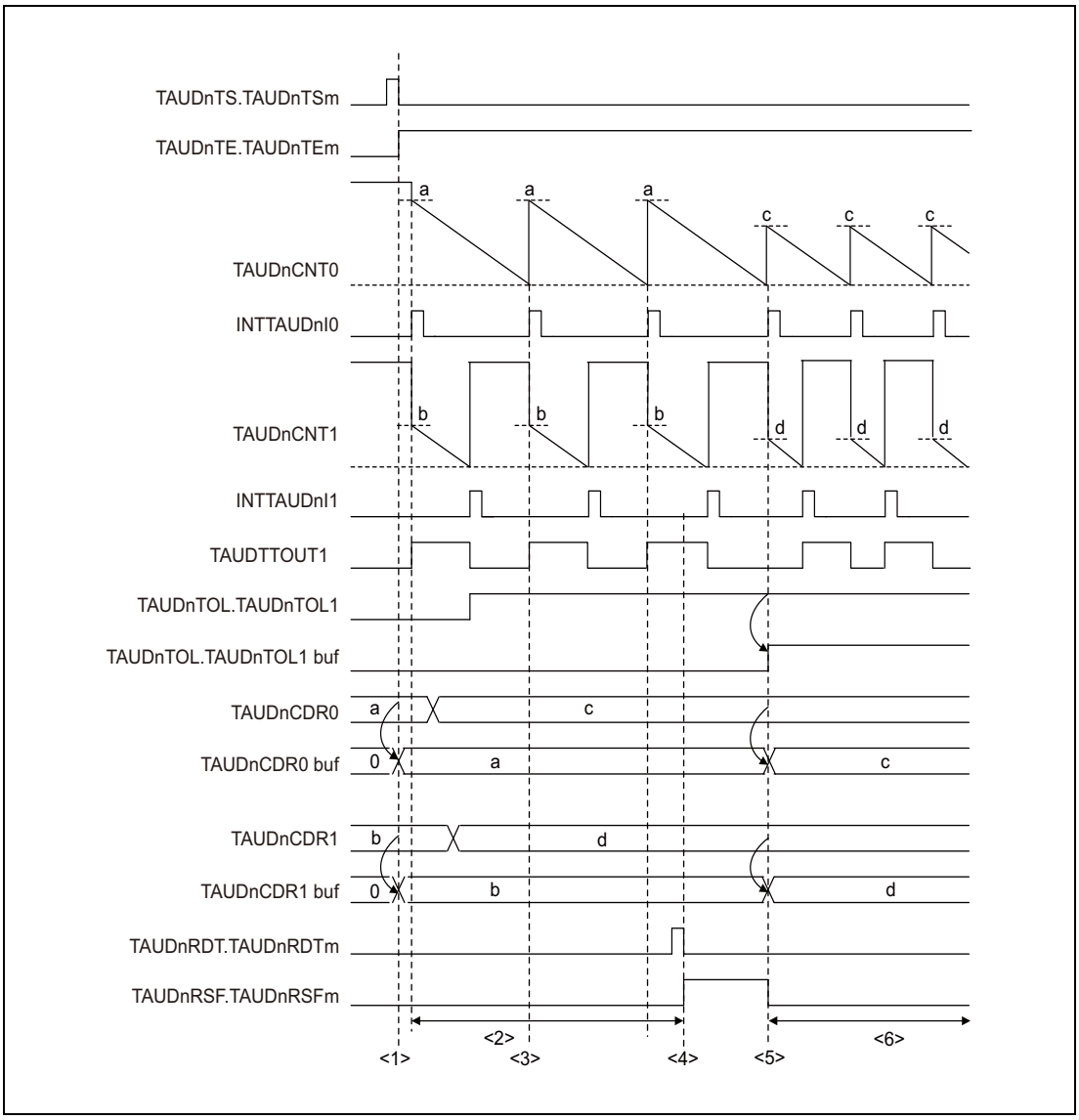



**Figure 25.6 Simultaneous Rewrite when the Master Channel (Re)starts Counting**

**Setting:**

CH0 is the master channel, which starts counting down, and CH1 represents an arbitrary slave channel. The simultaneous rewrite method A is applied.

**Description:**

(1) When TAUDnTS.TAUDnTSm is set to 1, TAUDnCDRm value is copied to the TAUDnCDRm buffer and TAUDnTOL.TAUDnTOLm value is copied to the TAUDnTOL.TAUDnTOLm buffer.

(2) The TAUDnCDRm and TAUDnTOL.TAUDnTOLm registers can be written at any time.

(3) CH0 restarts counting, but simultaneous rewrite does not occur because it is disabled (TAUDnRSF.TAUDnRSFm = 0)

(4) The reload data trigger bit (TAUDnRDT.TAUDnRDTm) is set to 1 which sets the status flag (TAUDnRSF.TAUDnRSFm = 1), enabling simultaneous rewrite.

(5) Because simultaneous rewrite is enabled, it is triggered when CH0 restarts counting. The TAUDnCDRm value is loaded into the TAUDnCDRm buffer and the TAUDnTOL.TAUDnTOLm value is loaded into the TAUDnTOL.TAUDnTOLm buffer.

(6) The counters count down and await the next simultaneous rewrite trigger. The values of TAUDnCDRm and TAUDnTOL.TAUDnTOLm can be changed again.

### 25.6.4.2 Simultaneous Rewrite at the Peak of a Triangular Wave of Slave Channel (Method B)

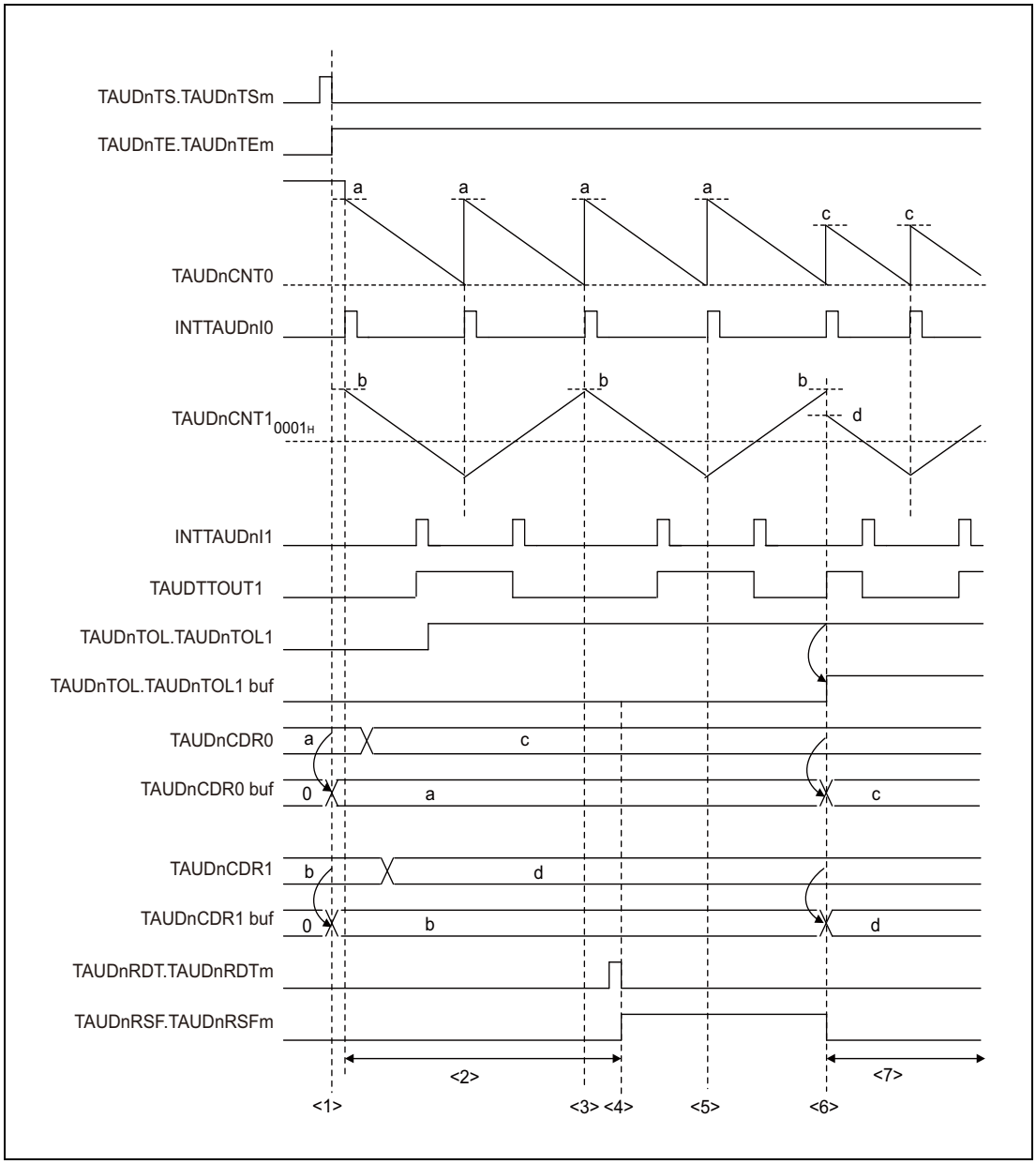



**Figure 25.7 Simultaneous Rewrite at the Peak of a Triangular Wave of Slave Channel**

**Setting:**

CH0 is the master channel which performs counting down, and CH1 represents an arbitrary slave channel. The simultaneous rewrite method B is applied.

**Description:**

(1) When TAUDnTS.TAUDnTSm is set to 1, TAUDnCDRm value is copied to the TAUDnCDRm buffer.

(2) The TAUDnCDRm and TAUDnTOL registers can be written at any time.

(3) Simultaneous rewrite does not occur because it is disabled (TAUDnRSF.TAUDnRSFm = 0).

(4) The reload data trigger bit (TAUDnRDT.TAUDnRDTm) is set to 1 which sets the status flag (TAUDnRSF.TAUDnRSFm = 1), enabling simultaneous rewrite.

(5) Simultaneous rewrite does not take place at the bottom of the triangular cycle.

(6) Simultaneous rewrite takes place at the top of the triangular cycle. The TAUDnCDRm value is loaded into the TAUDnCDRm buffer, the TAUDnTOL.TAUDnTOLm value is loaded into the TAUDnTOL.TAUDnTOLm buffer.

(7) The counters count down and await the next simultaneous rewrite trigger. The values of TAUDnCDRm and TAUDnTOL.TAUDnTOLm can be changed again.

#### 25.6.4.3 Simultaneous Rewrite when INTTAUDnIm is Generated on an Upper Channel Specified by TAUDnRDC.TAUDnRDCm (Method C1)

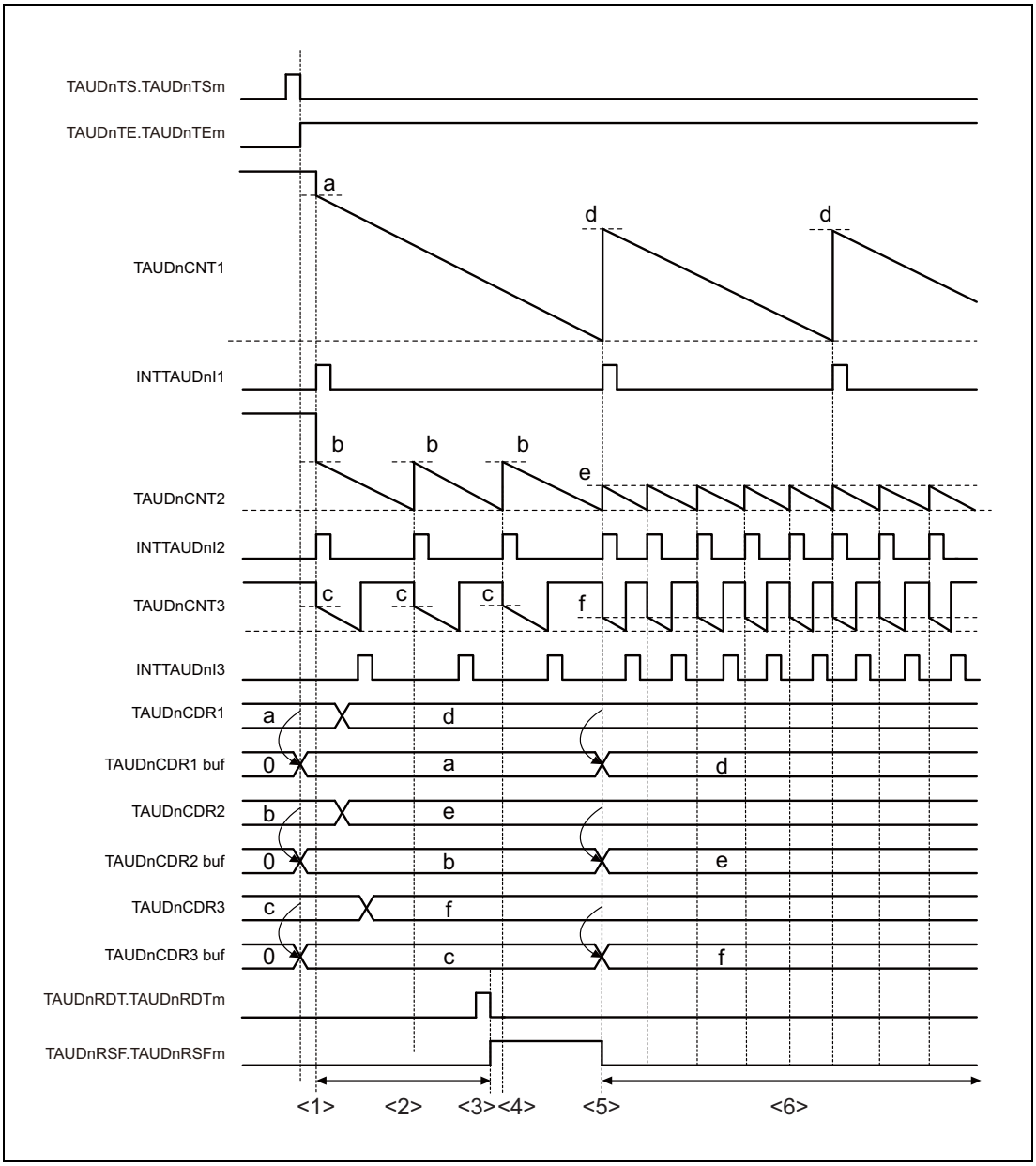



**Figure 25.8 Simultaneous Rewrite When INTTAUDnIm Is Generated on an Upper Channel Specified by TAUDnRDC.TAUDnRDCm**

**Setting:**

CH1 is an upper channel which performs counting down, CH2 is a master channel, and CH3 is the slave channel. The simultaneous rewrite method C1 is applied. The TAUDnRDC register specifies a channel which generates simultaneous rewrite triggers.

**Description:**

(1) When TAUDnTS.TAUDnTSm is set to 1, TAUDnCDRm value is copied to the TAUDnCDRm buffer.

(2) The TAUDnCDRm register is always ready to write.

(3) By setting the reload data trigger bit (TAUDnRDT.TAUDnRDTm) to 1, the status flag is set (TAUDnRSF.TAUDnRSFm = 1) to enable simultaneous rewrite.

(4) Simultaneous rewrite is triggered only by a CH1 interrupt. Therefore, simultaneous rewrite is not conducted even if enabled.

(5) Simultaneous rewrite is triggered by INT1 which is generated when counter 1 reaches $0000_H$. The TAUDnCDRm values are loaded into the corresponding TAUDnCDRm buffers.

(6) The counter counts down and awaits the next simultaneous rewrite trigger. The values of the TAUDnCDRm registers can be rechanged.

### 25.6.4.4 Simultaneous Rewrite when INTTAUDnIm is Generated on an Upper Channel Specified by TAUDnRDC.TAUDnRDCm that in Turn is Triggered by an External Signal (Method C2)

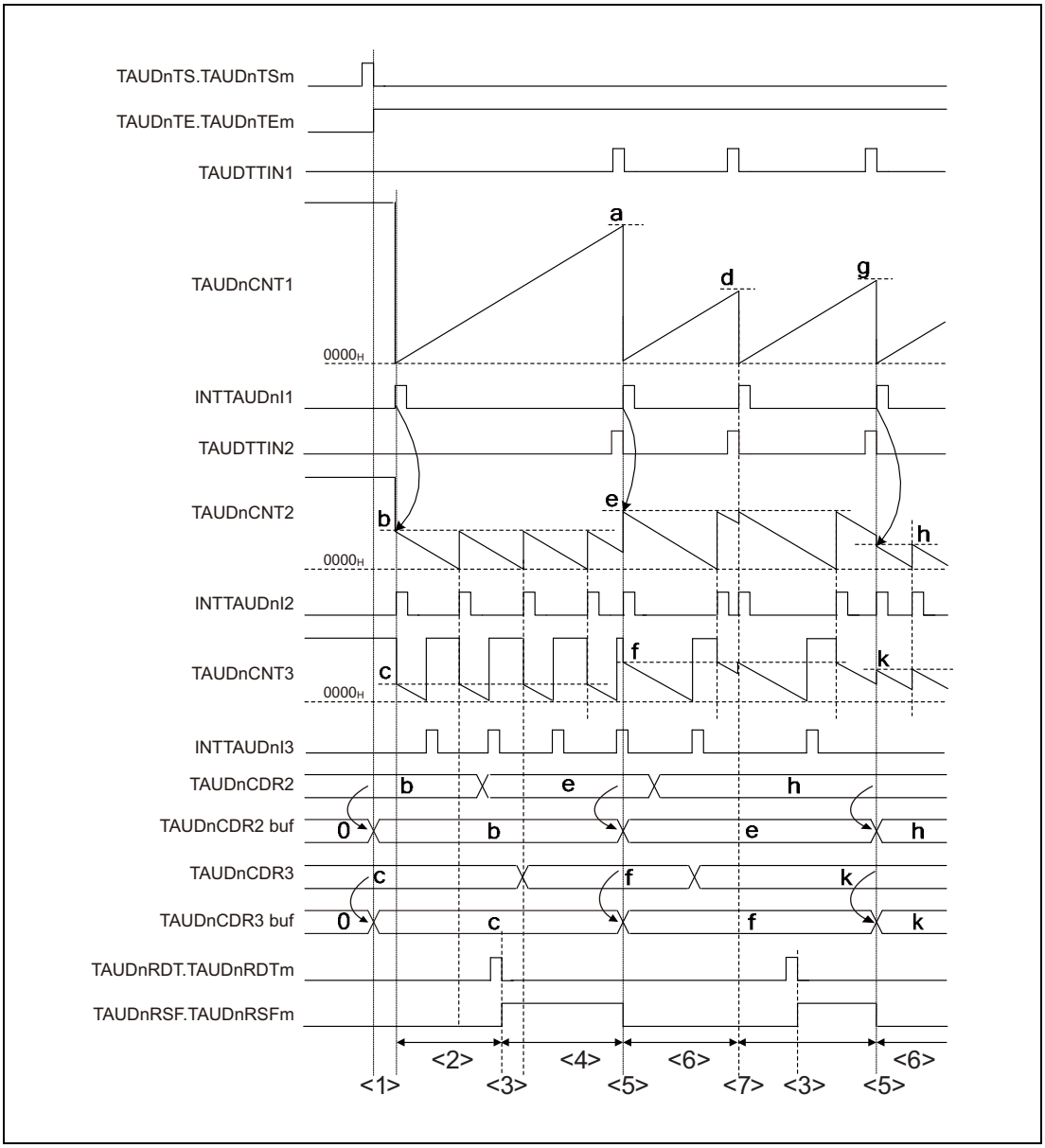



**Figure 25.9 Simultaneous Rewrite when INTTAUDnIm is Generated on an Upper Channel Specified by TAUDnRDC.TAUDnRDCm that in Turn is Triggered by an External Signal**

**Setting:**

CH1 is an upper channel which performs counting up, CH2 is a master channel, and CH3 is the slave channel. The synchronous channel operation method C2 is applied. The TAUDnRDC register specifies which upper channel is monitored for an INTTAUDnIm trigger.

**Description:**

(1) When TAUDnTS.TAUDnTSm is set to 1, TAUDnCDRm value is copied to the TAUDnCDRm buffer. However, as TAUDnCDR1 operates in capture mode, TAUDnCDR1 value is not copied to the TAUDnCDR1 buffer.

(2) The TAUDnCDRm register is always ready to write.

(3) By setting the reload data trigger bit (TAUDnRDT.TAUDnRDTm) to 1, the status flag is set (TAUDnRSF.TAUDnRSFm = 1) to enable simultaneous rewrite.

(4) Simultaneous rewrite is triggered only by a CH1 interrupt. Therefore, simultaneous rewrite is not conducted even if enabled.

(5) Simultaneous rewrite is triggered by INT1 which is caused by external signal TIN1. The TAUDnCDRm values are written to the corresponding TAUDnCDRm buffers.

(6) The counters count down and await the next simultaneous rewrite trigger. The values of the TAUDnCDRm registers can be changed again.

(7) An external signal occurs at TIN2 but simultaneous rewrite does not take place because it is disabled (TAUDnRSF.TAUDnRSFm = 0).

## 25.7 Channel Output Modes

The output of the TAUDTTOUTm pin can be controlled in two ways, the latter of which can be further split into individual modes.

- By software (TAUDnTOE.TAUDnTOEm = 0)
  When controlled by software, the value written in the output register bit (TAUDnTO.TAUDnTOm) is sent to the output pin (TAUDTTOUTm).
- By TAUD signals (TAUDnTOE.TAUDnTOEm = 1)
  When controlled by TAUD signals, the output level of TAUDTTOUTm is set or reset or toggled by internal signals. The value of TAUDnTO.TAUDnTOm is updated accordingly to reflect the value of TAUDTTOUTm.
  - Independently (TAUDnTOM.TAUDnTOMm = 0)
    In case of independent operation, the output of the TAUDTTOUTm pin is only affected by settings of channel m. Therefore, independent channel operation should be selected (TAUDnTOM.TAUDnTOMm = 0).
  - Synchronously (TAUDnTOM.TAUDnTOMm = 1)
    In case of synchronous operation, the output of the TAUDTTOUTm pin is affected by settings of channel m and those of other channels. Therefore, synchronous channel operation should be selected for all synchronized channels (TAUDnTOM.TAUDnTOMm = 1).

The TAUDnTO.TAUDnTOm bit can always be read to determine the current value of TAUDTTOUTm, regardless of whether the pin is controlled by software, operated independently, or operated synchronously.

### Control bits

The settings of the control bits required to select a specific channel output mode are listed in **Table 25.48, Channel Output Modes**.

The channel output modes are described in details below.

- **Section 25.7.2, Channel Output Modes Controlled Independently by TAUDn Signals**
- **Section 25.7.3, Channel Output Modes Controlled Synchronously by TAUDn Signals**

### Batch operation of TAUDnTOm bit

Whether a set value is reflected to the TAUDnTOm bit or not is controlled by the TAUDnTOE.TAUDnTOEm bit.

The TAUDnTOm setting is written only to the bit (channel) set with TAUDnTOE.TAUDnTOEm bit = 0 when a write to the TAUDnTO register is attempted. No TAUDnTOm setting is reflected to the bit (channel) set with TAUDnTOE.TAUDnTOEm bit = 1.

**NOTE**

TAUDnTO.TAUDnTOm bit is placed so that its bit number corresponds to a channel number.

### Output logic

Positive logic or negative logic of the output is specified by control bit TAUDnTOL.TAUDnTOLm.

The value of TAUDnTOL.TAUDnTOLm bit should be set before the counter is started. It can only be changed during operation with PWM output function or triangle PWM output function. Otherwise, changes to TAUDnTOL.TAUDnTOLm result in an undefined TAUDTTOUTm signal output.

See **Section 25.6, Simultaneous Rewrite**.

The various channel output modes and the channel output control bits are listed in **Table 25.48**.

**Table 25.48 Channel Output Modes**

| Channel Output Mode | TAUDn TOE. TAUDn TOEm | TAUDn TOM. TAUDn TOMm | TAUDn TOC. TAUDn TOCm | TAUDn TDE. TAUDn TDEm | TAUDn TRE. TAUDn TREm | TAUDn TME. TAUDn TMEm | TAUDn TDM. TAUDn TDMm |
|---|---|---|---|---|---|---|---|
| **By software** | | | | | | | |
| Independent channel output mode controlled by software | 0 | X | | | | | |
| **By TAUD signals, independently** | | | | | | | |
| Independent channel output mode 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| with real-time output | | | | | 1 | | |
| Independent channel output mode 2 | | | 1 | | 0 | | |
| **By TAUD signals, synchronously** | | | | | | | |
| Synchronous channel output mode 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| with non-complementary modulation output | | | | | 1 | X | |
| Synchronous channel output mode 2 | | | 1 | 0 | 0 | 0 | 0 |
| with dead time output | | | | 1 | | | |
| with one-phase PWM output | | | | | | | 1 |
| with complementary modulation output | | | | | 1 | 1 | 0 |
| with non-complementary modulation output | | | 1 | 0 | | | |

- All combinations not listed in this table are forbidden.
- Bits marked with an x can be set to any value.

**NOTES**

1. The following bits cannot be changed during count operation (TAUDnTE.TAUDnTEm = 1):
   - TAUDnTOM.TAUDnTOMm
   - TAUDnTOC.TAUDnTOCm
   - TAUDnTDE.TAUDnTDEm
   - TAUDnTRE.TAUDnTREm
   - TAUDnTDM.TAUDnTDMm
2. The following bits cannot be changed during count operation (TAUDnTE.TAUDnTEm = 1) except in channel output modes with modulation output:
   - TAUDnTME.TAUDnTMEm
   - TAUDnTDL.TAUDnTDLm

### 25.7.1 General Procedures for Specifying a Channel Output Mode

This section describes the general procedures for specifying a TAUDTTOUTm channel output mode. The prerequisite is that timer output operation is disabled (TAUDnTOE.TAUDnTOEm = 0).

(1) Set TAUDnTO.TAUDnTOm to specify the initial level of the TAUDTTOUTm output.

(2) Set channel output mode according to **Table 25.48, Channel Output Modes**, and the output logic using the TAUDnTOL.TAUDnTOLm bit.

(3) Start the counter (TAUDnTS.TAUDnTSm = 1).

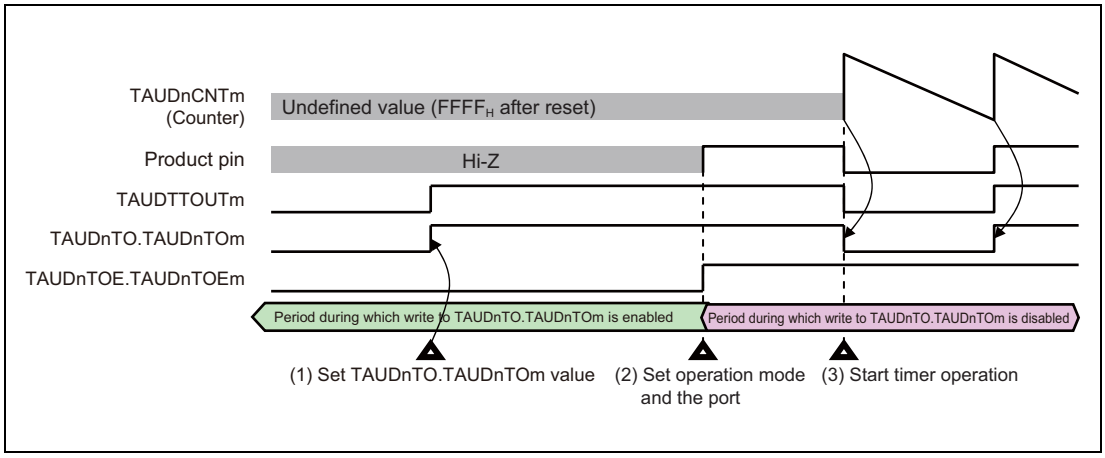



**Figure 25.10 General Procedure for Specifying a TAUDTTOUTm Channel Output Mode**

## 25.7.2 Channel Output Modes Controlled Independently by TAUDn Signals

This section lists the channel output modes that are controlled independently by TAUDn signals. The control bits used to specify a mode are listed in **Table 25.48, Channel Output Modes**.

### 25.7.2.1 Independent Channel Output Mode 1

#### Set/reset conditions

In this output mode, TAUDTTOUTm toggles when INTTAUDnIm is detected. The value of TAUDnTOL.TAUDnTOLm is ignored.

#### Prerequisites

There are no prerequisites other than those shown in **Table 25.48, Channel Output Modes**.

### 25.7.2.2 Independent Channel Output Mode 1 with Real-Time Output

In this output mode, the value of TAUDnTRO.TAUDnTROm bit of the trigger channel is output to TAUDTTOUTm. The trigger channel is specified by setting the corresponding TAUDnTRC.TAUDnTRCm bit to 1. It controls all lower channels for which TAUDnTRC.TAUDnTRCm = 0.

#### Set/reset conditions

The value of TAUDnTRO.TAUDnTROm bit is sent to TAUDTTOUTm only when an INTTAUDnIm interrupt occurs on the trigger channel. The interrupt is generated either:

- at certain specified intervals or
- on detection of a valid TAUDTTINm input edge/counter start

The type of trigger is set using the TAUDnCMORm.TAUDnMD[4:1] bits.

#### Prerequisites

Both the master and slave channels can be set as a trigger generation channel. A channel for which TAUDnTRC.TAUDnTRCm is set to 1 serves as a trigger generation channel regardless of the value of TAUDnTRE.TAUDnTREm.

If there is no channel for which TAUDnTRC.TAUDnTRCm is set to 1 or if TAUDnTRC.TAUDnTRC0 = 0, real-time output cannot take place.

This can be seen in **Figure 25.11**.

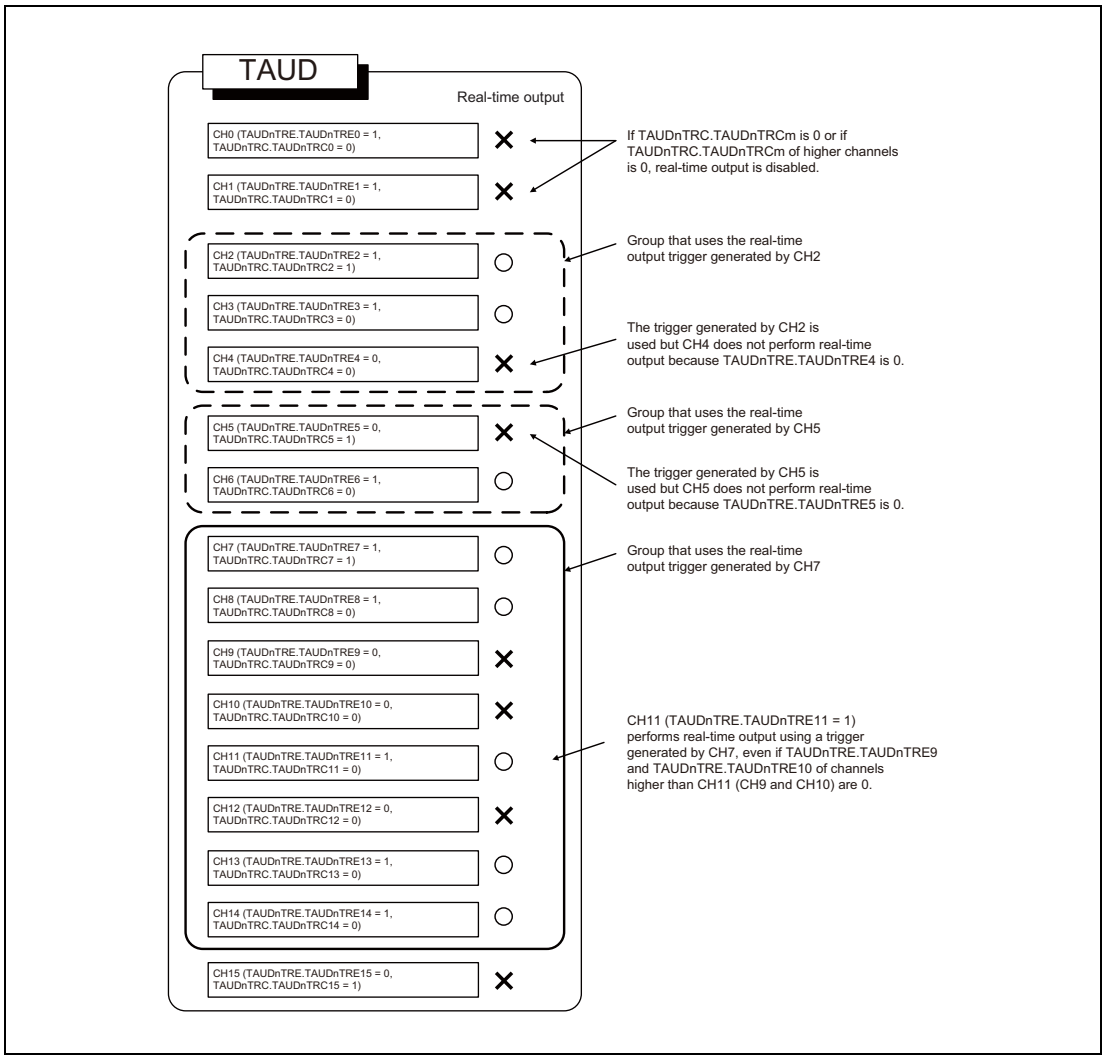



**Figure 25.11 Real-Time Output**

### 25.7.2.3 Independent Channel Output Mode 2

#### Set/reset conditions

In this output mode, TAUDTTOUTm is set when INTTAUDnIm occurs at the time of count start, and reset when INTTAUDnIm occurs due to a match between TAUDnCNTm and TAUDnCDRm.

#### Prerequisites

There are no prerequisites other than those shown in **Table 25.48, Channel Output Modes**.

## 25.7.3 Channel Output Modes Controlled Synchronously by TAUDn Signals

This section lists the channel output modes that are controlled synchronously by TAUDn signals. The control bits used to specify a mode are listed in **Table 25.48, Channel Output Modes**.

### 25.7.3.1 Synchronous Channel Output Mode 1

#### Set/reset conditions

In this output mode, INTTAUDnIm of master channel serves as a set signal and INTTAUDnIm of the slave channel as a reset signal. If INTTAUDnIm of master channel and INTTAUDnIm of the slave channel are generated at the same time, INTTAUDnIm of the slave channel (reset signal) has priority over INTTAUDnIm (set signal) of master channel, i.e., the master channel is ignored.

#### Prerequisites

There are no prerequisites other than those shown in **Table 25.48, Channel Output Modes**.

### 25.7.3.2 Synchronous Channel Output Mode 1 with Non-Complementary Modulation Output

#### Set/reset conditions

In this output mode, TAUDTTOUTm outputs the result of an AND operation between the PWM output and the real-time output bit (TAUDnTRO.TAUDnTROm) of a channel.

The phase period to which the dead time is added is specified using the TAUDnTDL.TAUDnTDLm bit; for positive phase set TAUDnTDL.TAUDnTDLm = 0 and for negative phase set TAUDnTDL.TAUDnTDLm = 1.

#### Prerequisites

A set of at least three channels is required to generate the PWM output. The master channel and slave channel 1 generate a period, and slave channel 2 generates the duty cycle. In typical applications, five more slave channels are also used that operate in the same manner as slave channel 2.

Only the PWM output and the real-time output bit of the same channel can be combined.

TAUDnTRO.TAUDnTROm, TAUDnTME.TAUDnTMEm, and TAUDnTDL.TAUDnTDLm can only be changed during count operation.

- If TAUDnTME.TAUDnTMEm is changed, its new value is applied upon detection of INTTAUDnIm on the specified channel.
- If TAUDnTME.TAUDnTMEm and TAUDnTDL.TAUDnTDLm are changed, their new values are applied upon detection of INTTAUDnIm on the master channel.

### 25.7.3.3 Synchronous Channel Output Mode 2

In this output mode, the operating mode should be set to count-up/-down mode. The result is a triangle PWM output at TAUDTTOUTm. For details, see **Section 25.15.7, Triangle PWM Output Function**.

#### Set/reset conditions

TAUDnCNTm of the slave channel counts down and up alternatively. When it passes $0001_H$ it generates an interrupt, causing TAUDTTOUTm to toggle.

**Prerequisites**

A set of two channels is required to generate the triangle PWM output. TAUDTTOUTm should be set to 0 before the function starts.

### 25.7.3.4 Synchronous Channel Output Mode 2 with Dead Time Output

In this output mode, a dead time delay is added to TAUDTTOUTm. The set/reset conditions are shown in **Figure 25.12**.

**Set/reset conditions**

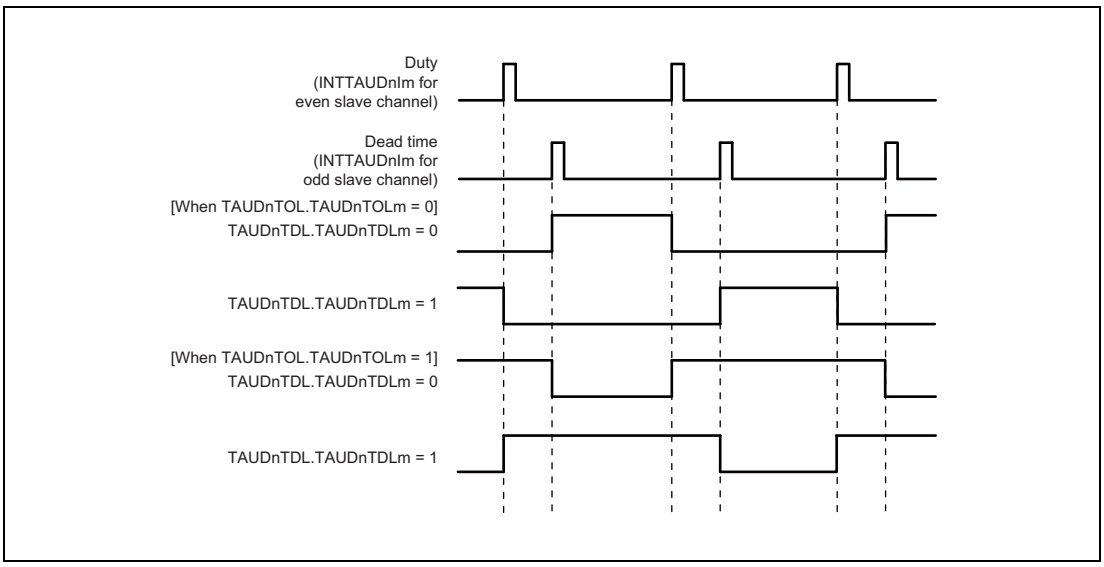



**Figure 25.12 Set/Reset Conditions for Synchronous Channel Output Mode 2 with Dead Time Output**

With regard to the edge to which dead time is added, set TAUDnTDL.TAUDnTDLm = 0 for rising edges and TAUDnTDL.TAUDnTDLm = 1 for falling edges.

**Prerequisites**

Dead time control requires a set of three channels, each operating in the following modes:

- One master channel
  The master channel should be set to interval timer mode.
- One even slave channel
  The even slave channel should be set to count-up/-down mode.
- One odd slave channel (even channel + 1)
  The odd slave channel should be set to one-count mode.

The values of the following bits should be the same for the odd channel and the even channel:

- TAUDnTOE.TAUDnTOEm
- TAUDnTME.TAUDnTMEm
- TAUDnTRE.TAUDnTREm
- TAUDnTOM.TAUDnTOMm
- TAUDnTOC.TAUDnTOCm

- TAUDnTDE.TAUDnTDEm
- TAUDnTDM.TAUDnTDMm

### 25.7.3.5 Synchronous Channel Output Mode 2 with One-Phase PWM Output

In this output mode, a dead time delay is added to TAUDTTOUTm. The set/reset conditions are shown in **Figure 25.13**.

#### Set/reset conditions

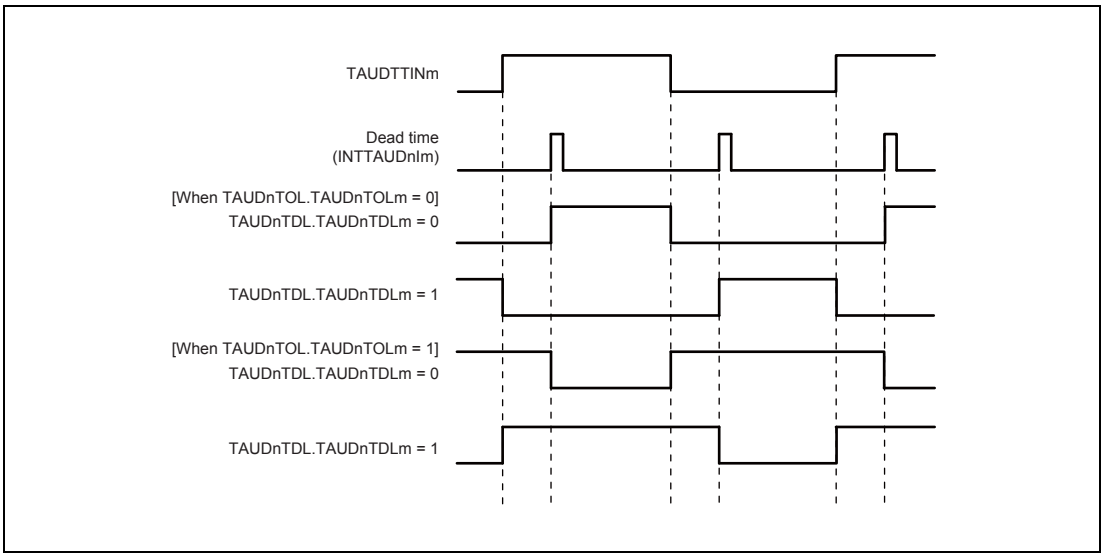



**Figure 25.13 Set/Reset Conditions for Synchronous Channel Output Mode 2 with One-Phase PWM Output**

With regard to the edge to which dead time is added, set TAUDnTDL.TAUDnTDLm = 0 for rising edges and TAUDnTDL.TAUDnTDLm = 1 for falling edges.

#### Prerequisites

One-phase PWM output control requires a set of two channels:

- One even slave channel
- One odd slave channel (even channel + 1)
  The odd slave channel should be set to one-count mode.

The values of the following bits should be the same for the odd channel and the even channel:

- TAUDnTOE.TAUDnTOEm
- TAUDnTME.TAUDnTMEm
- TAUDnTRE.TAUDnTREm
- TAUDnTOM.TAUDnTOMm
- TAUDnTOC.TAUDnTOCm
- TAUDnTDE.TAUDnTDEm
- TAUDnTDM.TAUDnTDMm

### 25.7.3.6 Synchronous Channel Output Mode 2 with Complementary Modulation Output

#### Set/reset conditions

In this output mode, TAUDTTOUTm outputs a PWM signal, a high signal, or a low signal depending on the value of real-time output bit (TAUDnTRO.TAUDnTROm), the modulation output bit (TAUDnTME.TAUDnTMEm), and the output level bit (TAUDnTOL.TAUDnTOLm) of a pair of slave channels.

For details, see **Section 25.16.3, Complementary Modulation Output Function**.

#### Prerequisites

A set of at least four channels is required for this mode. The master channel and slave channel 1 generate a period, slave channel 2 generates a duty cycle, and slave channel 3 generates dead time. Slave channels 2 and 3 are a pair. In typical applications, four more channels are also used, which operates in the same manner as slave channels 2 and 3 respectively.

TAUDnTRO.TAUDnTROm, TAUDnTME.TAUDnTMEm, and TAUDnTDL.TAUDnTDLm can only be changed during count operation.

- If TAUDnTME.TAUDnTMEm is changed during operation, its new value is applied upon detection of INTTAUDnIm at the specified channel.
- If TAUDnTME.TAUDnTMEm and TAUDnTDL.TAUDnTDLm are changed, their new values are applied upon detection of INTTAUDnIm on an even slave channel.

### 25.7.3.7 Synchronous Channel Output Mode 2 with Non-Complementary Modulation Output

The difference from synchronous channel output mode 1 with non-complementary modulation output is the PWM wave shape.

Mode 1 has a square wave while mode 2 has a triangular wave.

## 25.8 Start Timing in Each Operating Modes

This section describes the timing at which the counter starts after TAUDnTS.TAUDnTSm is set to 1 in each operating mode.

In all modes, the value of data register and whether or not an interrupt occurs depends on mode and register settings.

**CAUTION**

**The count start timing described in this section is for your reference. Actually, the count start timing depends on the count clock timing.**

### 25.8.1 Interval Timer Mode, Judge Mode, Capture Mode, Count-up/-down Mode, and Count Capture Mode

The counter starts operating with the next count clock cycle after TAUDnTS.TAUDnTSm is set to 1. The value of data register is also loaded when the counter starts.

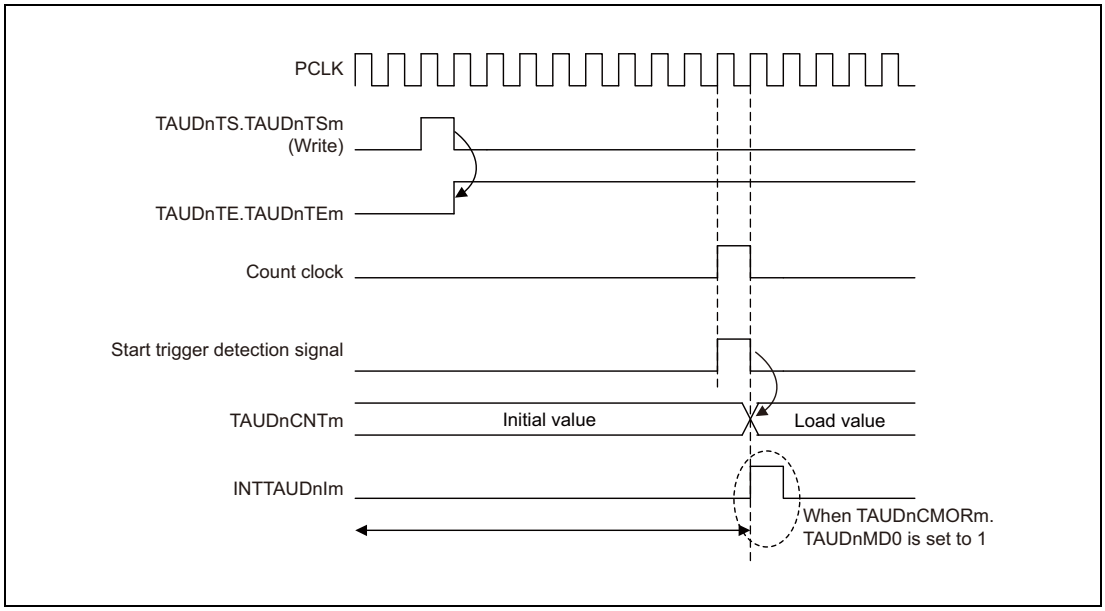



**Figure 25.14 Start Timing in Interval Timer Mode, Judge Mode, Capture Mode, Count-up/-down Mode, and Count Capture Mode**

**NOTE**

Make sure to set TAUDnCMORm.TAUDnMD0 to 0 when using the count-up/-down mode.

### 25.8.2 Event Count Mode

The value of data register is loaded as soon as TAUDnTS.TAUDnTSm is set to 1. The counter also starts immediately. The value of data register decrements when the subsequent count clock cycle starts.

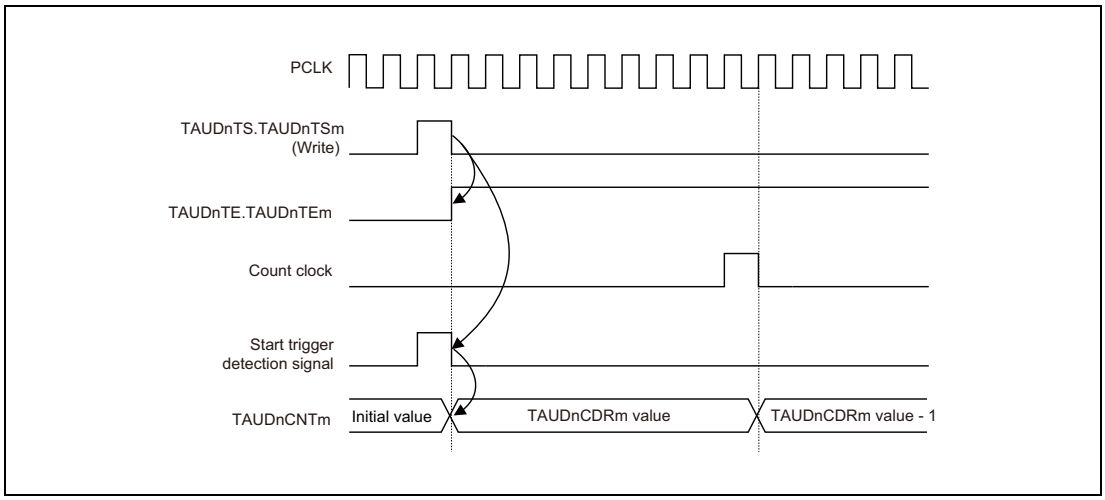



**Figure 25.15 Start Timing in Event Count Mode**

### 25.8.3 Other Operating Modes

In other operating modes, the counter operation start timing is triggered only upon detection of a valid edge of TAUDTTINm. Once the counter starts, the value of data register is also loaded. The count clock cycles, which are irrelevant to start of counter operation, determine the frequency with which all operations take place.

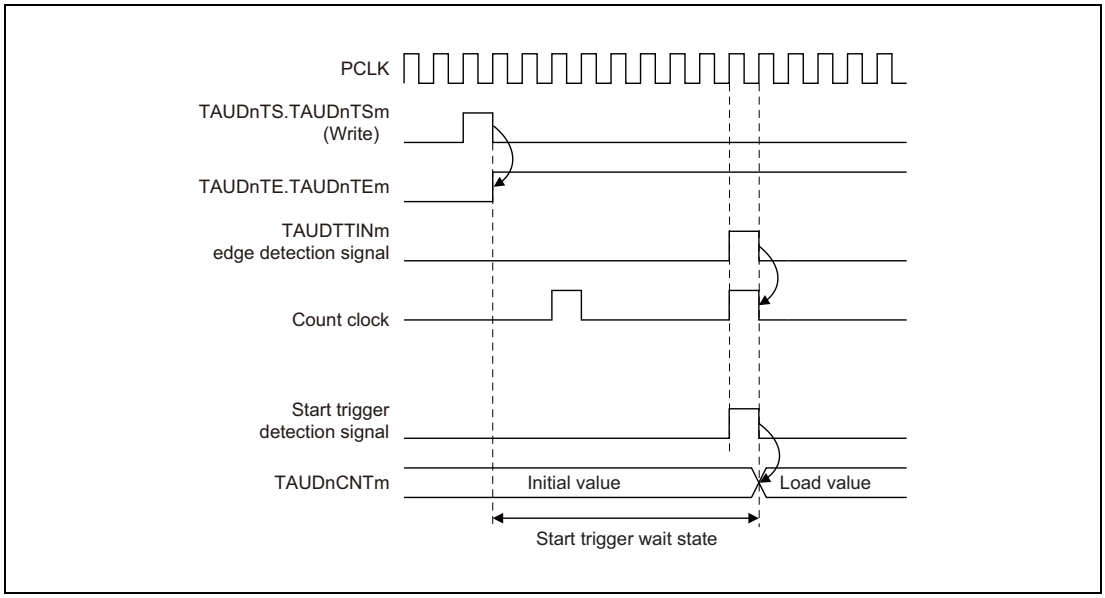



**Figure 25.16 Count Start Timing in Other Operating Modes**

## 25.9 TAUDTTOUTm Output and INTTAUDnIm Generation when Counter Starts or Restarts

When the counter starts, it is possible to specify whether an INTTAUDnIm is generated using the TAUDnCMORm.TAUDnMD0 bit. The generation of INTTAUDnIm when the TAUDnCMORm.TAUDnMD0 bit starts counting and the effect to TAUDTTOUTm depend on the selected function. For details, refer to the description of TAUDnCMORm.TAUDnMD0 of each function.

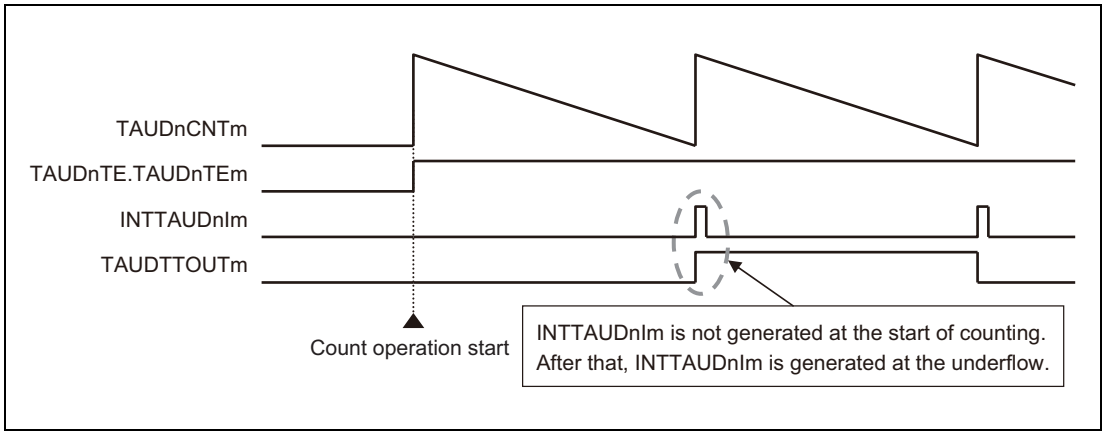



**Figure 25.17 INTTAUDnIm Generation Timing (when TAUDnCMORm.TAUDnMD0 = 0)**

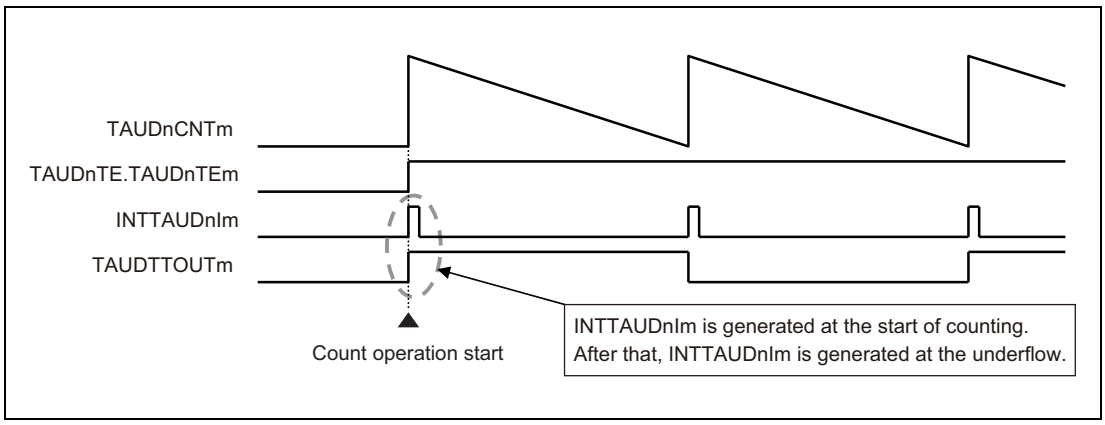



**Figure 25.18 INTTAUDnIm Generation Timing (when TAUDnCMORm.TAUDnMD0 = 1)**

## 25.10 Interrupt Generation upon Overflow

In certain independent functions, an interrupt is not generated when the counter value reaches $FFFF_H$ and an overflow occurs during count-up. This section describes how to generate an interrupt by combining channel operation in a mode that counts up and in a mode that counts down.

The appropriate operation mode for the second channel depends on the operation mode of the first channel. Nevertheless, the principle is the same for all combinations:

- Find an operation mode for the second channel that counts down in such a manner, that it reaches $0000_H$ at the same time as the first channel overflows (TAUDnCNTm = $FFFF_H$).
- Set TAUDnCDRm of the second channel to $FFFF_H$.
- The two channels must count at the same speed (i.e. they must have the same count clock).
- Both channels are triggered by the same TAUDTTINm input.
- The trigger detection settings (TAUDnCMORm.TAUDnSTS[2:0] and TAUDnCMURm.TAUDnTIS[1:0]) must be identical for both channels.

**Result:**

The down-counter of the second channel reaches $0000_H$ at exactly the same time as the up-counter of the first channel overflows (TAUDnCNTm = $FFFF_H$). Thus the second channel generates the desired interrupt.

The following sections list the operating modes that count down that are required to match specific operating modes that count up, as well as example timing diagrams.

### 25.10.1 Combination of the TAUDTTINm Input Pulse Interval Measurement Function and the TAUDTTINm Input Interval Timer Function

When the capture trigger is input simultaneously to TAUDTTINm of both channels, INTTAUDnIm of the TAUDTTINm input interval timer function can detect the overflow when TAUDnCNTm of the TAUDTTINm input pulse interval measurement function exceeds $FFFF_H$.

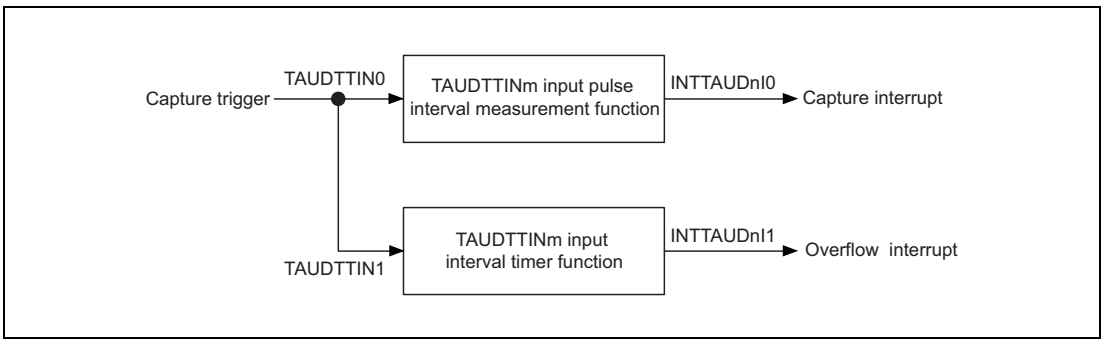



**Figure 25.19 Combination of the TAUDTTINm Input Pulse Interval Measurement Function and the TAUDTTINm Input Interval Timer Function**

**Timing diagram**

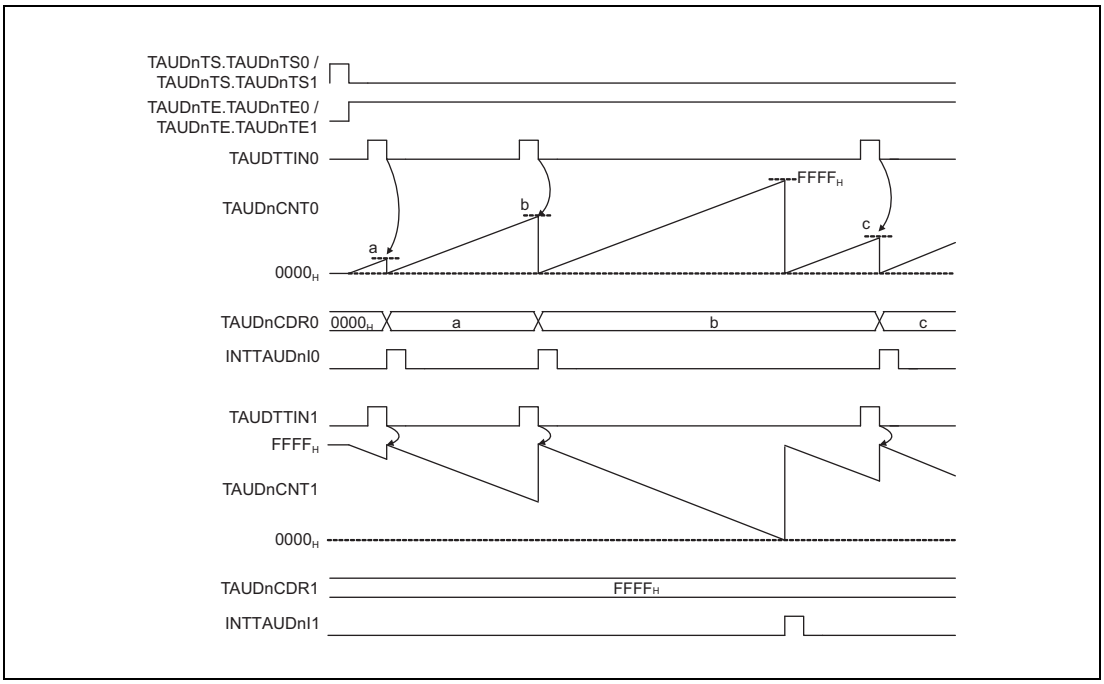



**Figure 25.20 Interrupt Generation via Combination of the TAUDTTINm Input Pulse Interval Measurement Function and the TAUDTTINm Input Interval Timer Function**

### 25.10.2 Combination of the TAUDTTINm Input Signal Width Measurement Function and the Overflow Interrupt Output Function (at Measuring the TAUDTTINm Width)

When the capture trigger is input simultaneously to TAUDTTINm of both channels, INTTAUDnIm of the overflow interrupt output function (at measuring the TAUDTTINm width) can detect the overflow when TAUDnCNTm of the TAUDTTINm input signal width measurement function exceeds $FFFF_H$.

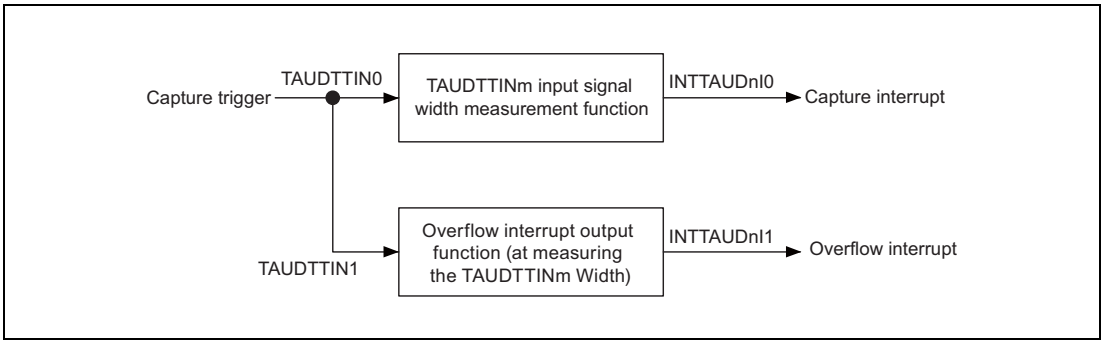



**Figure 25.21 Combination of the TAUDTTINm Input Signal Width Measurement Function and the Overflow Interrupt Output Function (at Measuring the TAUDTTINm Width)**

#### Timing diagram

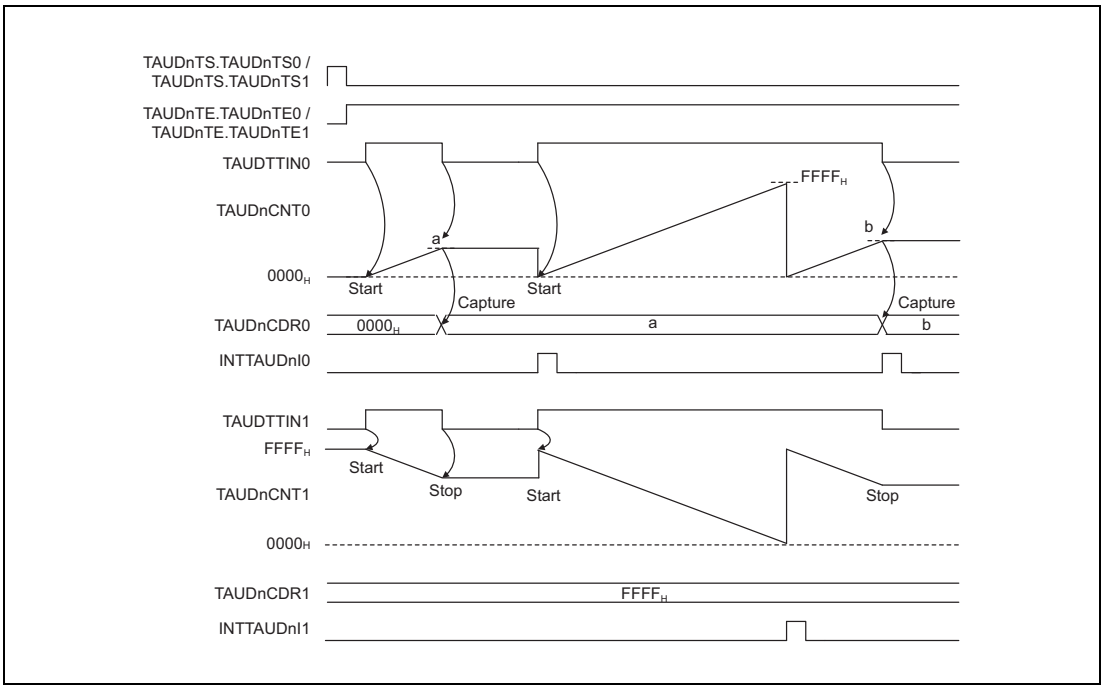



**Figure 25.22 Interrupt Generation via Combination of the TAUDTTINm Input Signal Width Measurement Function and the Overflow Interrupt Output Function (at Measuring the TAUDTTINm Width)**

### 25.10.3 Combination of the TAUDTTINm Input Position Detection Function and the Interval Timer Function

When the counters of both channels are enabled simultaneously, INTTAUDnIm of the interval timer function can detect the overflow when TAUDnCNTm of the TAUDTTINm input position detection function exceeds $FFFF_H$.

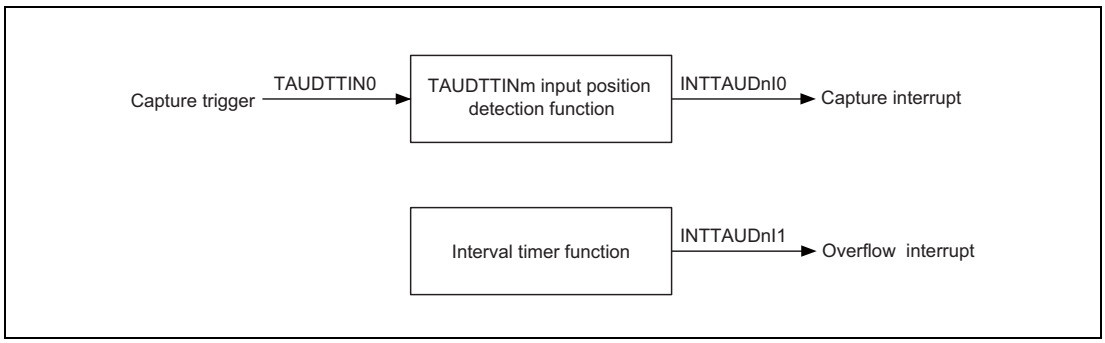



**Figure 25.23 Combination of the TAUDTTINm Input Position Detection Function and the Interval Timer Function**

#### Timing diagram

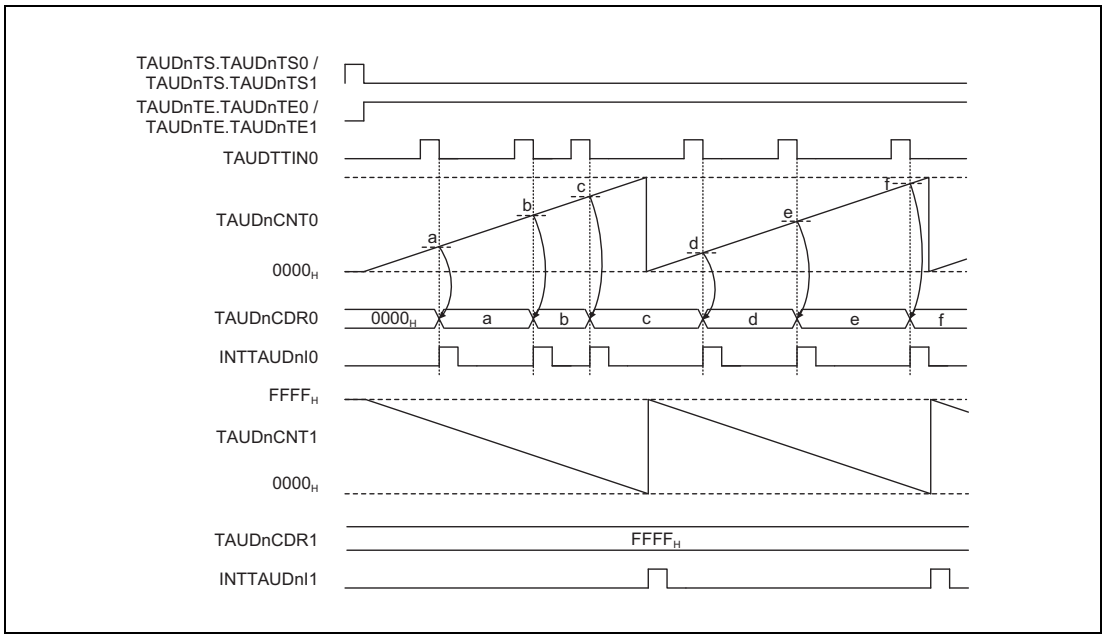



**Figure 25.24 Interrupt Generation via Combination of the TAUDTTINm Input Position Detection Function and the Interval Timer Function**

### 25.10.4 Combination of the TAUDTTINm Input Period Count Detection Function and the Overflow Interrupt Output Function (at Detecting the TAUDTTINm Input Period Count)

When the capture trigger is input simultaneously to TAUDTTINm of both channels, INTTAUDnIm of the overflow interrupt output function (at detecting the TAUDTTINm input period count) can detect the overflow when TAUDnCNTm of the TAUDTTINm input period count detection function exceeds $FFFF_H$.

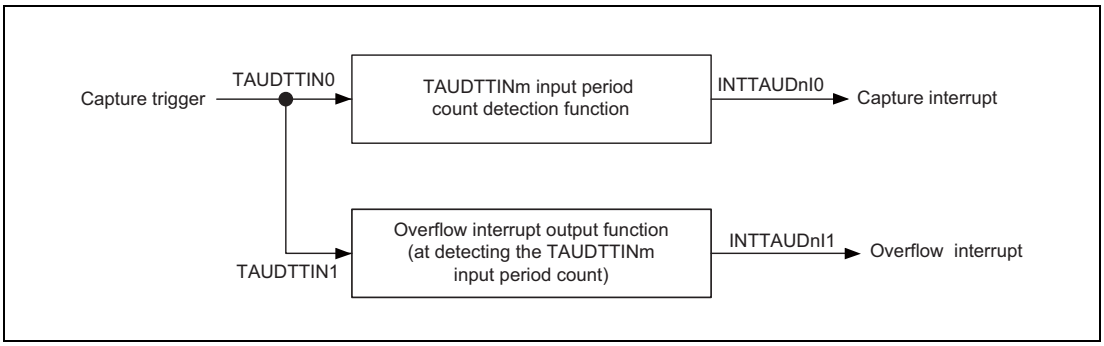



**Figure 25.25 Combination of the TAUDTTINm Input Period Count Detection Function and the Overflow Interrupt Output Function (at Detecting the TAUDTTINm Input Period Count)**

#### Timing diagram

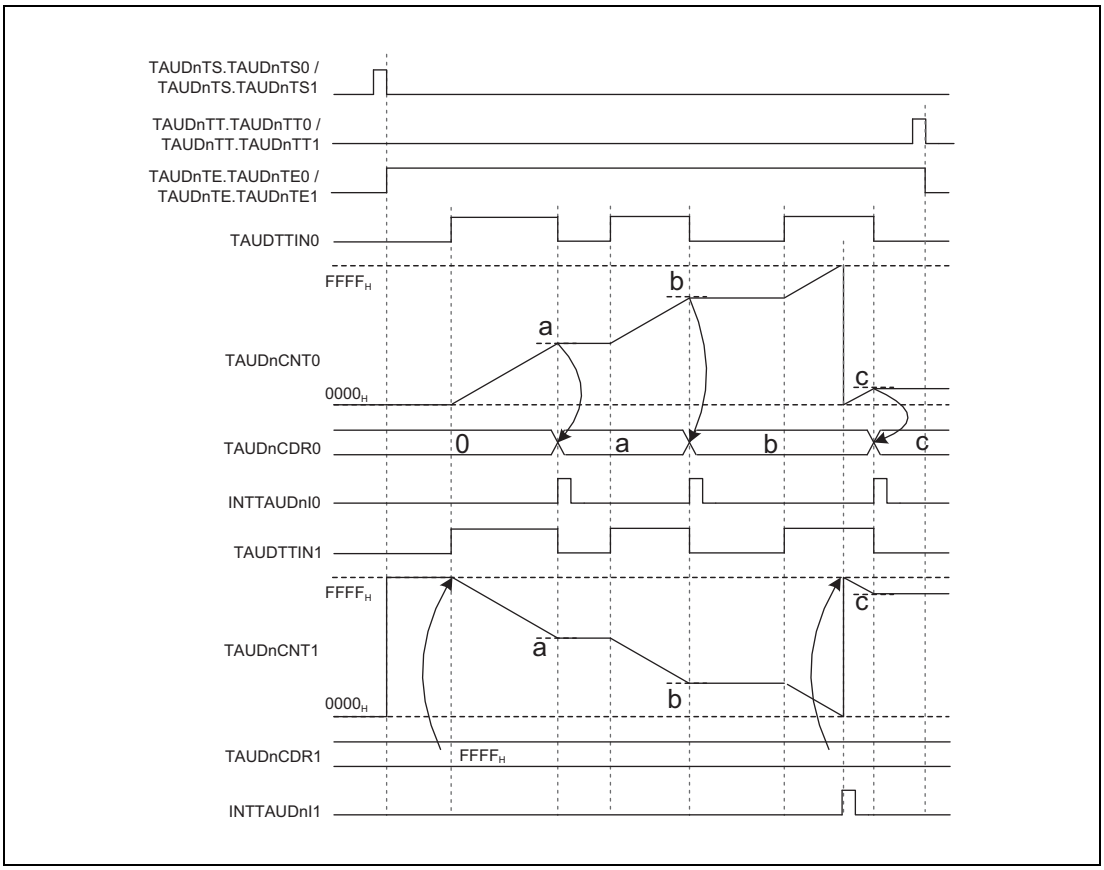



**Figure 25.26 Interrupt Generation via Combination of the TAUDTTINm Input Period Count Detection Function and the Overflow Interrupt Output Function (at Detecting the TAUDTTINm Input Period Count)**

## 25.11 TAUDTTINm Edge Detection

Edge detection is based on the operation clock. This means that an edge can only be detected at the next rising edge of the operation clock. This can lead to a maximum delay of one operation clock cycle.

**Figure 25.27** shows when edge detection takes place.

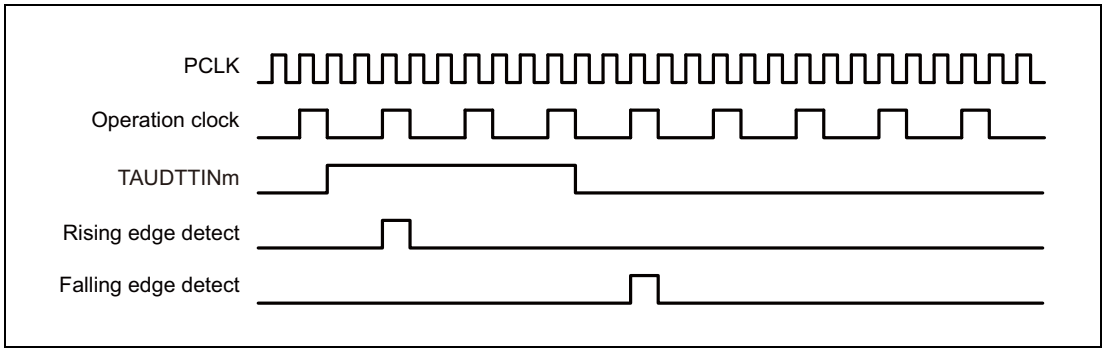



**Figure 25.27 Basic Edge Detection Timing**

**Figure 25.27** shows an operation timing image. Actually, a noise filter or synchronization circuit which is located between the TAUDnIm pin and TAUDn causes a delay time.

## 25.12 Independent Channel Operation Functions

The following sections list the independent channel operation functions provided by the Timer Array Unit D. For a general overview of independent channel operation, see **Section 25.2, Overview**.

This section describes functions that generate interrupts at regular intervals or with a specified delay.

### 25.12.1 Interval Timer Function

#### 25.12.1.1 Overview

**Summary**

This function is used as a reference timer for generating timer interrupts (INTTAUDnIm) at regular intervals. When an interrupt is generated, the TAUDTTOUTm signal toggles, resulting in a square wave.

**Prerequisites**

- The operation mode must be set to Interval Timer Mode, see **Table 25.49, Contents of the TAUDnCMORm Register for Interval Timer Function**.
- The channel output mode must be set to Independent Channel Output Mode 1, see **Section 25.7, Channel Output Modes**.

**Functional description**

The counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This in turn sets TAUDnTE.TAUDnTEm = 1, enabling count operation. The current value of TAUDnCDRm is written to TAUDnCNTm and the counter starts to count down from this value.

When the counter reaches $0000_H$, INTTAUDnIm is generated and the TAUDTTOUTm signal toggles. TAUDnCNTm then reloads the TAUDnCDRm value and subsequently continues operation.

The value of TAUDnCDRm can be rewritten at any time, and the changed value of TAUDnCDRm is applied the next time the counter starts to count down.

The counter can be stopped by setting TAUDnTT.TAUDnTTm to 1, which in turn sets TAUDnTE.TAUDnTEm to 0. TAUDnCNTm and TAUDTTOUTm stop but retain their values. The counter can be restarted by setting TAUDnTS.TAUDnTSm to 1. The counter can also be forcibly restarted (without stopping it first) by setting TAUDnTS.TAUDnTSm to 1 during operation.

**Conditions**

If the TAUDnCMORm.TAUDnMD0 bit is set to 0, the first interrupt after a start or restart is not generated, and therefore TAUDTTOUTm does not toggle. This results in a negative TAUDTTOUTm signal compared to when TAUDnCMORm.TAUDnMD0 is set to 1. For details see **Section 25.9, TAUDTTOUTm Output and INTTAUDnIm Generation when Counter Starts or Restarts**.

#### 25.12.1.2 Equations

INTTAUDnIm cycle = count clock cycle × (TAUDnCDRm + 1)

TAUDTTOUTm square wave cycle = count clock cycle × (TAUDnCDRm + 1) × 2

### 25.12.1.3 Block Diagram and General Timing Diagram

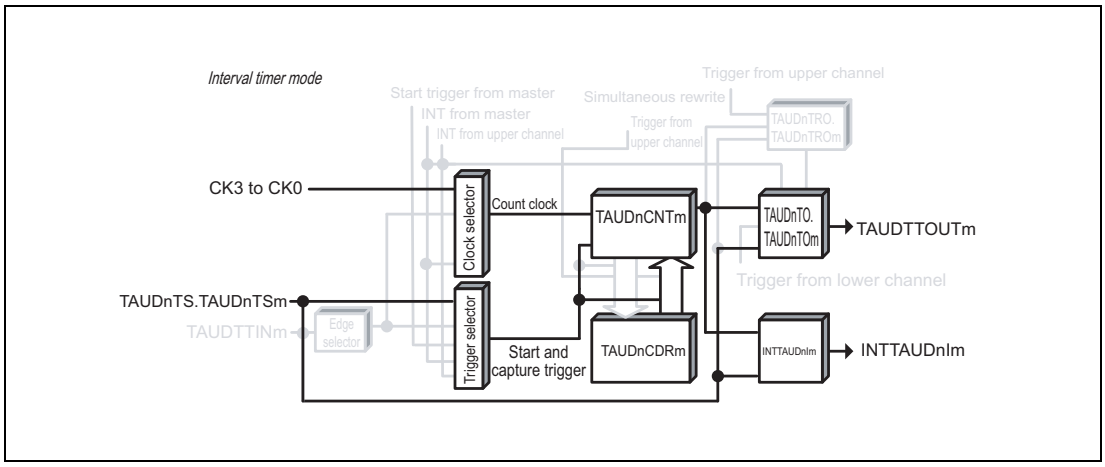



**Figure 25.28 Block Diagram of Interval Timer Function**

The following settings apply to the general timing diagram.

- INTTAUDnIm is generated at the beginning of operation (TAUDnCMORm.TAUDnMD0 = 1).

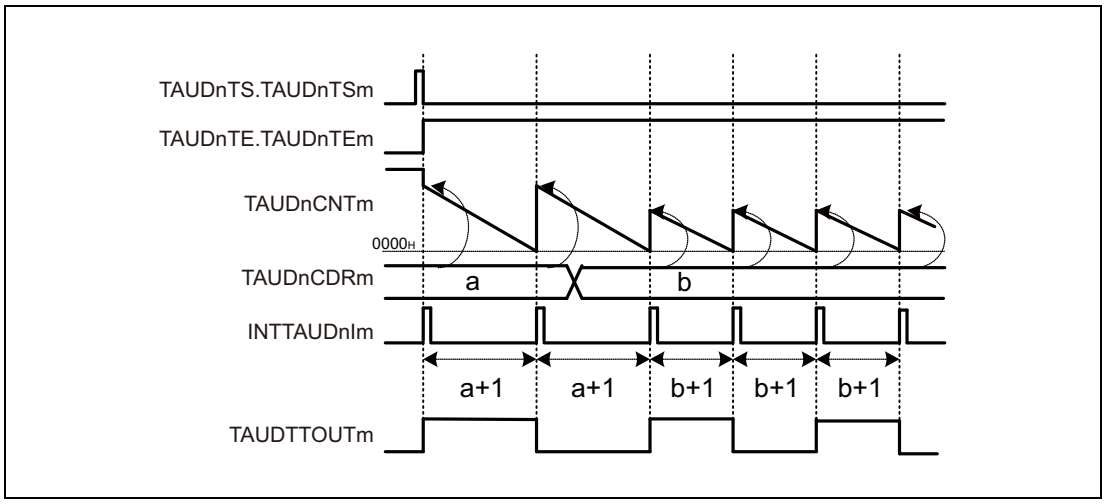



**Figure 25.29 General Timing Diagram of Interval Timer Function**

### 25.12.1.4 Register Settings

#### (1) TAUDnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS [1:0] | | TAUDnCCS [1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS [1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.49 Contents of the TAUDnCMORm Register for Interval Timer Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS [1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation, set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 000: Triggers the counter by software. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0000: Interval timer mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm is not generated to toggle TAUDTTOUTm at the beginning of an operation.<br>1: INTTAUDnIm is generated to toggle TAUDTTOUTm at the beginning of an operation. |

#### (2) TAUDnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.50 Contents of the TAUDnCMURm Register for Interval Timer Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode

Table 25.51 Control Bit Settings in Independent Channel Output Mode 1

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 0: Independent channel output |
| TAUDnTOC.TAUDnTOCm | 0: Operating mode 1 (Toggle mode if TAUDnTOM.TAUDnTOMm = 0) |
| TAUDnTOL.TAUDnTOLm | 0: The setting is disabled in toggle mode. (The value after reset.) |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set these bits to 0 |
| TAUDnTRC.TAUDnTRCm | |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

**NOTE**

The channel output mode can also be set to Channel Output Mode Controlled by Software by setting TAUDnTOE.TAUDnTOEm = 0. TAUDTTOUTm can then be controlled independently of the interrupts. For details, see **Section 25.7, Channel Output Modes**.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the interval timer function. Therefore, these registers should be set to 0.

Table 25.52 Simultaneous Rewrite Settings for Interval Timer Function

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

### 25.12.1.5 Operating Procedure for Interval Timer Function

Table 25.53 Operating Procedure for Interval Timer Function

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers as described in **Table 25.49, Contents of the TAUDnCMORm Register for Interval Timer Function**, and **Table 25.50, Contents of the TAUDnCMURm Register for Interval Timer Function**.<br><br>Set the value of TAUDnCDRm register.<br><br>Set channel output mode by setting the control bits as described in **Table 25.51, Control Bit Settings in Independent Channel Output Mode 1**. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is set to 1 and the counter starts.<br>The TAUDnCDRm value is loaded in TAUDnCNTm. When TAUDnCMORm.TAUDnMD0 = 1, INTTAUDnIm is generated and TAUDTTOUTm toggles. |
| During Operation | The TAUDnCDRm register value can be changed at any time.<br>The TAUDnCNTm register can be read at all times. | TAUDnCNTm counts down. When the counter reaches $0000_H$:<br>• The TAUDnCDRm value is loaded in TAUDnCNTm again and count operation continues.<br>• INTTAUDnIm is generated and TAUDTTOUTm toggles. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm and TAUDTTOUTm stop and retain their current values. |

Restart operation

### 25.12.1.6 Specific Timing Diagrams

#### (1) TAUDnCDRm = $0000_H$, count clock = PCLK/2

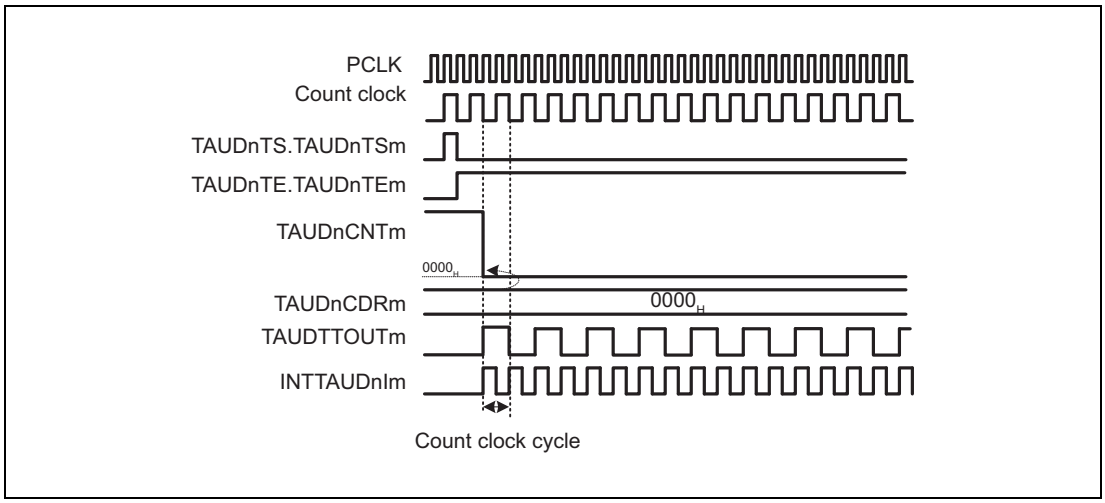



**Figure 25.30 TAUDnCDRm = $0000_H$, Count Clock = PCLK/2**

- If TAUDnCDRm = $0000_H$ and the count clock = PCLK/2, the TAUDnCDRm value is loaded into TAUDnCNTm every count clock, meaning that TAUDnCNTm is always $0000_H$.
- INTTAUDnIm is generated every count clock, resulting in TAUDTTOUTm toggling every count clock.

### (2) TAUDnCDRm = $0000_H$, count clock = PCLK

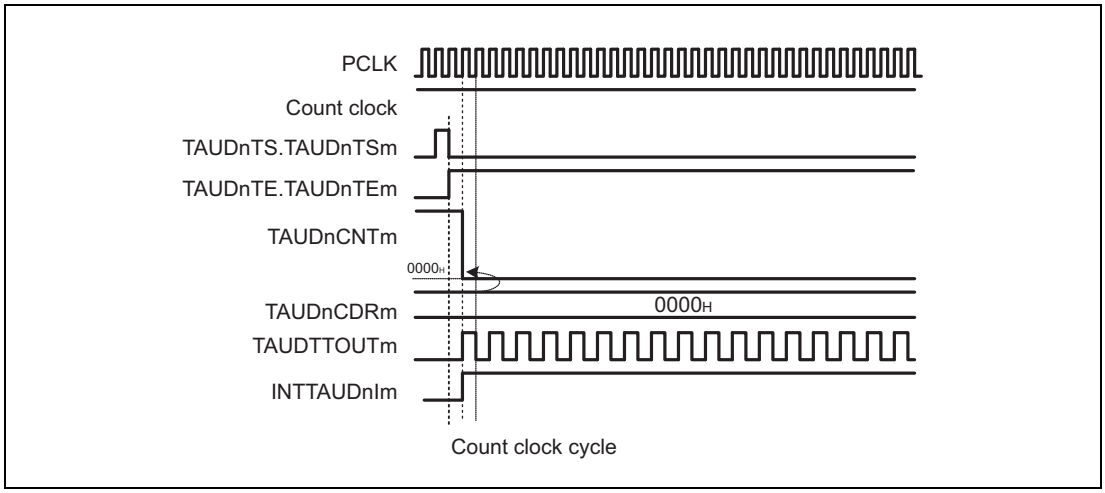



Figure 25.31 TAUDnCDRm = $0000_H$, Count Clock = PCLK

- If TAUDnCDRm = $0000_H$ and the count clock = PCLK, the TAUDnCDRm value is loaded into TAUDnCNTm every PCLK clock, meaning that TAUDnCNTm is always $0000_H$.
- INTTAUDnIm is fixed to the high level. Though the first interrupt is generated, subsequent interrupts are not generated.
  TAUDTTOUTm is toggled every PCLK clock.

### (3) Operation stop and restart

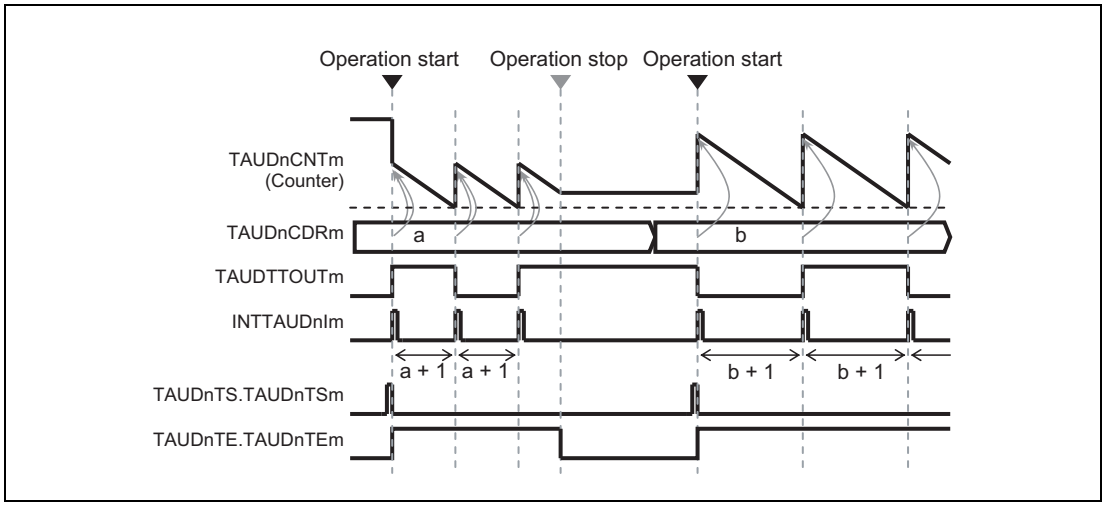



Figure 25.32 Operation Stop and Restart (TAUDnCMORm.TAUDnMD0 = 1)

- The counter can be stopped by setting TAUDnTT.TAUDnTTm to 1. This sets TAUDnTE.TAUDnTEm to 0.
- TAUDnCNTm and TAUDTTOUTm stop but retain their values.
- The counter can be restarted by setting TAUDnTS.TAUDnTSm to 1.

### (4) Forced restart (TAUDnCMORm.TAUDnMD0 = 1)

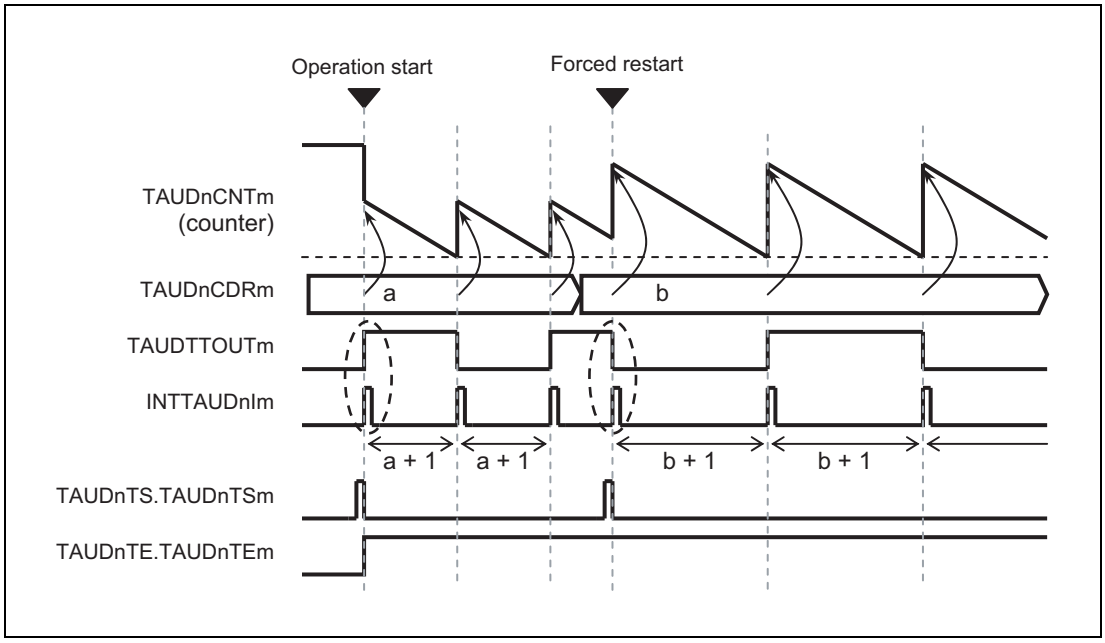



**Figure 25.33 Forced Restart Operation (TAUDnCMORm.TAUDnMD0 = 1)**

- The counter can be forcibly restarted (without stopping it first) by setting TAUDnTS.TAUDnTSm to 1 during operation.
- If the TAUDnCMORm.TAUDnMD0 bit is set to 1, the first interrupt after a start or restart is generated.
- When a forced restart is made, the TAUDnCDRm value is reflected to TAUDnCNTm and counting starts. Execute a forced restart to reflect the changed TAUDnCDRm value immediately.
- When a forced restart is made, an interrupt (INTTAUDnIm) is generated and TAUDTTOUTm is inverted.

### (5) Forced restart (TAUDnCMORm.TAUDnMD0 = 0)

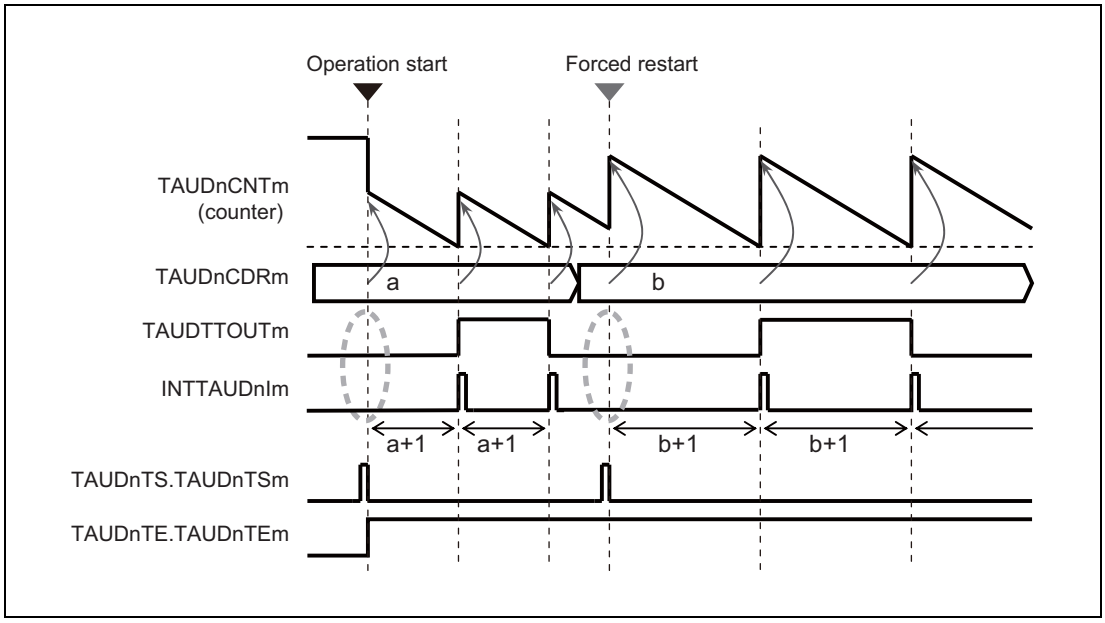



**Figure 25.34 Forced Restart Operation (TAUDnCMORm.TAUDnMD0 = 0)**

- When a forced restart is made, an interrupt (INTTAUDnIm) is not generated and TAUDTTOUTm is not inverted.

## 25.12.2 TAUDTTINm Input Interval Timer Function

### 25.12.2.1 Overview

#### Summary

This function is used as a reference timer for generating timer interrupts (INTTAUDnIm) at regular intervals or when a valid TAUDTTINm input edge is detected. When an interrupt is generated, the TAUDTTOUTm signal toggles, resulting in a square wave.

#### Prerequisites

- The operating mode should be set to interval timer mode. See **Table 25.54, Contents of the TAUDnCMORm Register for TAUDTTINm Input Interval Timer Function**.
- The channel output mode should be set to independent channel output mode 1. See **Section 25.7, Channel Output Modes**.

#### Functional description

This function operates in an identical manner to the interval timer function (see **Section 25.12.1, Interval Timer Function**) except that this function is restarted by a valid TAUDTTINm input edge. The type of edge used as a trigger is specified using the TAUDnCMURm.TAUDnTIS[1:0] bits. Either rising edge, falling edge, or rising and falling edges can be selected.

### 25.12.2.2 Equations

INTTAUDnIm cycle = count clock cycle × (TAUDnCDRm + 1)

TAUDTTOUTm square wave cycle = count clock cycle × (TAUDnCDRm + 1) × 2

### 25.12.2.3 Block Diagram and General Timing Diagram

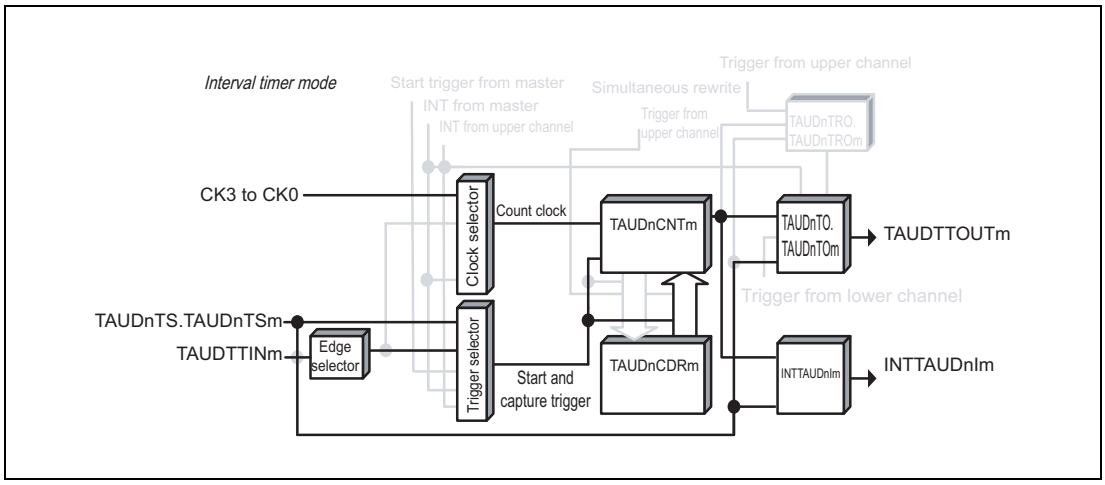



**Figure 25.35 Block Diagram of TAUDTTINm Input Interval Timer Function**

The following settings apply to the general timing diagram.

- INTTAUDnIm is generated at the beginning of operation (TAUDnCMORm.TAUDnMD0 = 1).
- Rising edge detection (TAUDnCMURm.TAUDnTIS[1:0] = $01_B$)

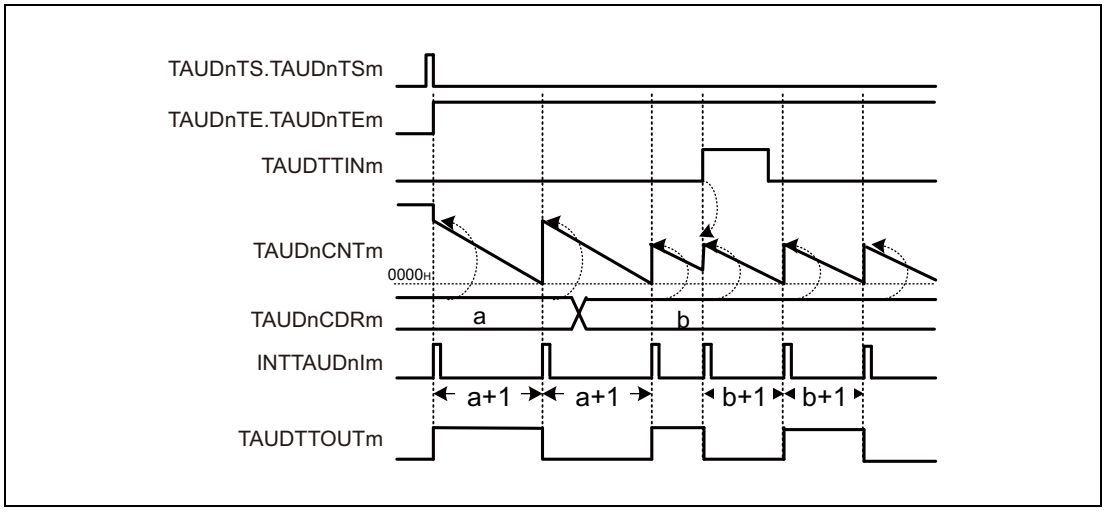



**Figure 25.36 General Timing Diagram of TAUDTTINm Input Interval Timer Function**

### 25.12.2.4 Register Settings

#### (1) TAUDnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS [1:0] | | TAUDnCCS [1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS [1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.54 Contents of the TAUDnCMORm Register for TAUDTTINm Input Interval Timer Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS [1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation. Set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 001: Valid TAUDTTINm input edge signal is used as an external start trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0000: Interval timer mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm is not generated to toggle TAUDTTOUTm at the beginning of an operation.<br>1: INTTAUDnIm is generated to toggle TAUDTTOUTm at the beginning of an operation. |

#### (2) TAUDnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.55 Contents of the TAUDnCMURm Register for TAUDTTINm Input Interval Timer Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Detection of falling edge<br>01: Detection of rising edge<br>10: Detection of rising and falling edges<br>11: Setting prohibited |

### (3) Channel output mode

Table 25.56 Control Bit Settings in Independent Channel Output Mode 1

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 0: Independent channel output |
| TAUDnTOC.TAUDnTOCm | 0: Operating mode 1<br>(Toggle mode if TAUDnTOM.TAUDnTOMm = 0) |
| TAUDnTOL.TAUDnTOLm | 0: The setting is disabled in toggle mode. (The value after reset.) |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled<br>(TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled<br>(TAUDnTRE.TAUDnTREm = 0), set these bits to 0 |
| TAUDnTRC.TAUDnTRCm | |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

NOTE

The channel output mode can also be set to Channel Output Mode Controlled by Software by setting TAUDnTOE.TAUDnTOEm = 0. TAUDTTOUTm can then be controlled independently of the interrupts. For details, see **Section 25.7, Channel Output Modes**.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the TAUDTTINm Input Interval Timer Function. Therefore, these registers should be set to 0.

Table 25.57 Simultaneous Rewrite Settings for TAUDTTINm Input Interval Timer Function

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled<br>(TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

### 25.12.2.5 Operating Procedure for TAUDTTINm Input Interval Timer Function

Table 25.58 Operating Procedure for TAUDTTINm Input Interval Timer Function

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers as described in **Table 25.54, Contents of the TAUDnCMORm Register for TAUDTTINm Input Interval Timer Function**, and **Table 25.55, Contents of the TAUDnCMURm Register for TAUDTTINm Input Interval Timer Function**.<br><br>Set the value of TAUDnCDRm register.<br><br>Set channel output mode by setting the control bits as described in **Table 25.56, Control Bit Settings in Independent Channel Output Mode 1**. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is set to 1 and the counter starts.<br>The TAUDnCDRm value is loaded in TAUDnCNTm.<br>When TAUDnCMORm.TAUDnMD0 = 1, INTTAUDnIm is generated and TAUDTTOUTm toggles. |
| During Operation | The values of the TAUDnCMURm.TAUDnTIS[1:0] and the TAUDnCDRm register are changeable at any time.<br>The TAUDnCNTm register can be read at all times.<br><br>Detection of TAUDTTINm edge | TAUDnCNTm counts down. When the counter reaches $0000_H$:<br>• The TAUDnCDRm value is loaded in TAUDnCNTm again and count operation continues.<br>• INTTAUDnIm is generated and TAUDTTOUTm toggles.<br>When a TAUDTTINm input valid edge is detected during count operation, the TAUDnCDRm value is loaded in TAUDnCNTm and count operation continues.<br>Afterwards, this procedure is repeated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm and TAUDTTOUTm stop and retain their current values. |

Restart Operation

### 25.12.2.6 Specific Timing Diagrams

The timing diagrams in **Section 25.12.1, Interval Timer Function** apply, and in addition the counter can also be restarted by a valid TAUDTTINm input edge.

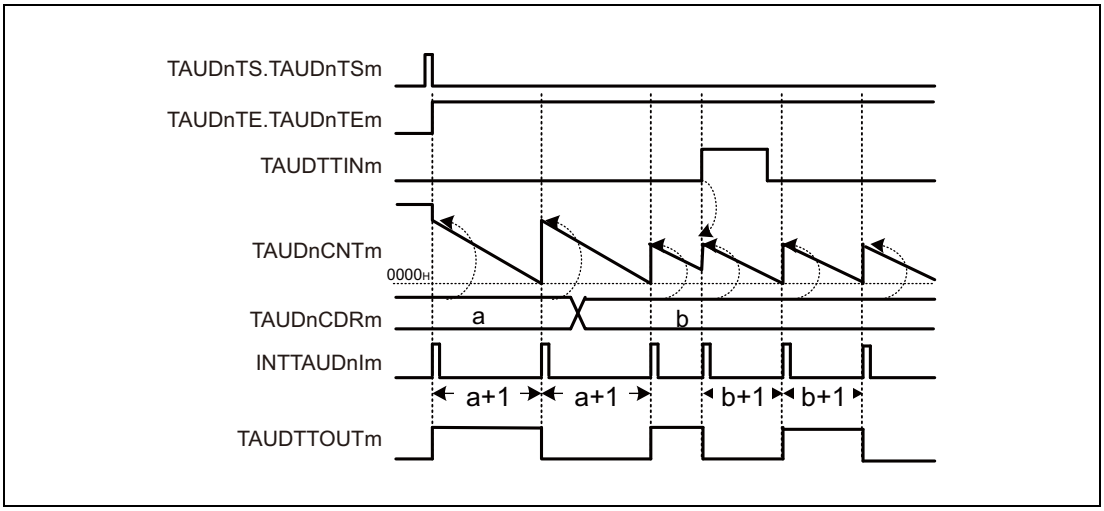



**Figure 25.37 Counter Triggered by Rising TAUDTTINm Input Edge (TAUDnCMURm.TAUDnTIS[1:0] = $01_B$), TAUDnCMORm.TAUDnMD0 = 1**

- If a valid TAUDTTINm input edge is detected, an interrupt is generated which causes TAUDTTOUTm to toggle. In this example, the valid edge is a rising edge (TAUDnCMURm.TAUDnTIS[1:0] = $01_B$)

## 25.12.3 Clock Divide Function

### 25.12.3.1 Overview

#### Summary

This function is used as a frequency divider. The frequency of the input signal TAUDTTINm is divided by a factor related to TAUDnCDRm, and the resulting signal is output to TAUDTTOUTm.

#### Prerequisites

- TAUDTTINm should have a fixed frequency.
- The operating mode should be set to interval timer mode. (See **Table 25.59, Contents of the TAUDnCMORm Register for Clock Divide Function**.)
- The channel output mode should be set to independent channel output mode 1. (See **Section 25.7, Channel Output Modes**.)

#### Functional description

The counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This in turn sets TAUDnTE.TAUDnTEm = 1, enabling count operation. The current value of TAUDnCDRm is loaded into TAUDnCNTm and the counter starts to count down from this value, using TAUDTTINm as a count clock.

When the counter value reaches $0000_H$, INTTAUDnIm occurs and TAUDTTOUTm signal is toggled. Then, TAUDnCDRm value is loaded into TAUDnCNTm to continue operation subsequently.

The value of TAUDnCDRm can be rewritten at any time. The changed value of TAUDnCDRm is applied when the counter starts to count down next time.

The counter can be stopped by setting TAUDnTT.TAUDnTTm = 1. This sets TAUDnTE.TAUDnTEm = 0. TAUDnCNTm and TAUDTTOUTm stop but retain their values. The function can be restarted by setting TAUDnTS.TAUDnTSm = 1. The counter can also be forcibly restarted without making a stop by setting TAUDnTS.TAUDnTSm = 1 during operation (forced restart).

#### Conditions

If the TAUDnCMORm.TAUDnMD0 bit is set to 0, the first interrupt after a start or restart is not generated, and therefore TAUDTTOUTm does not toggle. This results in a negative TAUDTTOUTm signal compared to when TAUDnCMORm.TAUDnMD0 is set to 1. For details, see **Section 25.9, TAUDTTOUTm Output and INTTAUDnIm Generation when Counter Starts or Restarts**.

**NOTE**

TAUDTTINm input signals are sampled at the frequency of the operation clock set by TAUDnCMORm.TAUDnCKS[1:0] bits. Therefore, the TAUDTTOUTm output clock cycle has an error of ± 1 operation clock cycle.

### 25.12.3.2 Equations

- When rising edge detection is selected:
  TAUDTTOUTm frequency = TAUDTTINm frequency/[(TAUDnCDRm + 1) × 2]
- When falling edge detection is selected:
  TAUDTTOUTm frequency = TAUDTTINm frequency/[(TAUDnCDRm + 1) × 2]
- When falling and rising edge detection is selected:
  TAUDTTOUTm frequency = TAUDTTINm frequency/(TAUDnCDRm + 1)

### 25.12.3.3 Block Diagram and General Timing Diagram

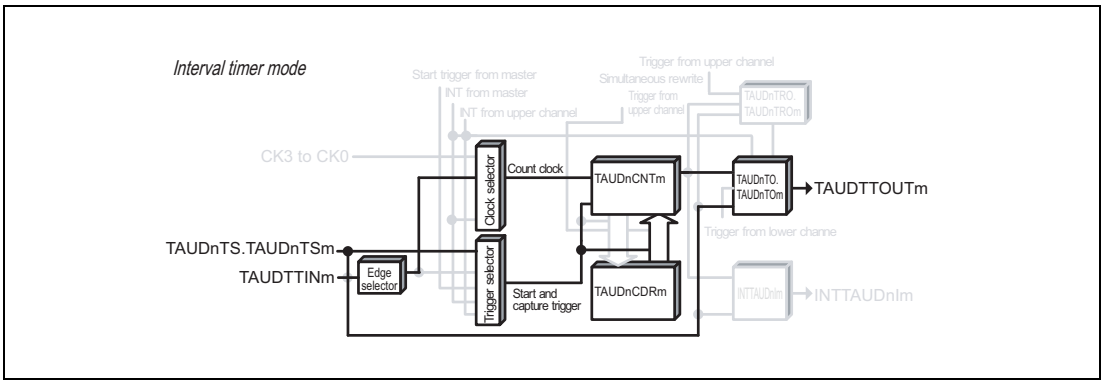



**Figure 25.38 Block Diagram of Clock Divide Function**

The following settings apply to the general timing diagram.

- INTTAUDnIm is generated at the beginning of operation. (TAUDnCMORm.TAUDnMD0 = 1)
- Detection of rising edge (TAUDnCMURm.TAUDnTIS[1:0] = $01_B$)

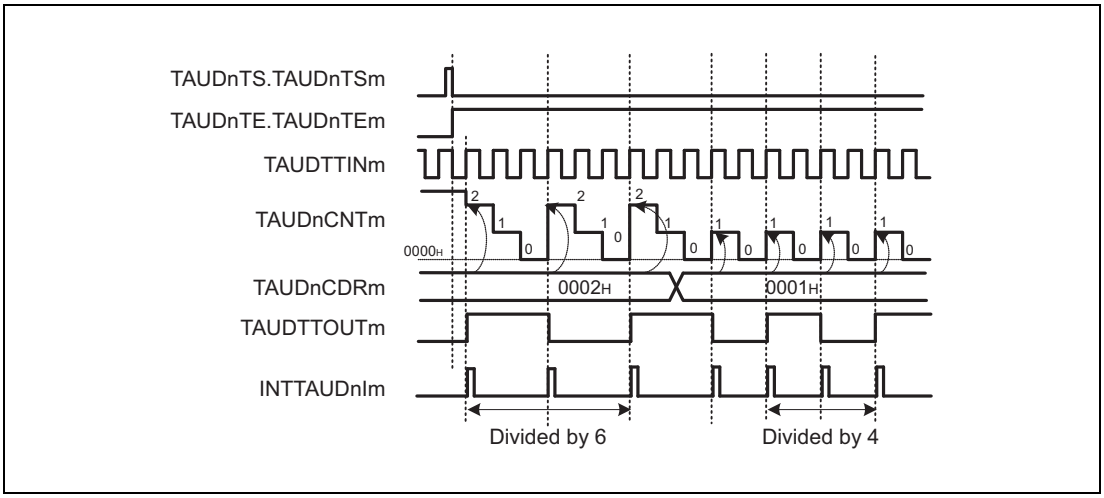



**Figure 25.39 General Timing Diagram of Clock Divide Function**

### 25.12.3.4 Register Settings

#### (1) TAUDnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS [1:0] | | TAUDnCCS [1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS [1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.59 Contents of the TAUDnCMORm Register for Clock Divide Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 01: Valid TAUDTTINm input edge is used as a count clock. |
| 11 | TAUDnMAS | 0: Independent operation. Set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 000: Trigger the counter using software. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0000: Interval timer mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm is not generated to toggle TAUDTTOUTm at the beginning of an operation.<br>1: INTTAUDnIm is generated and TAUDTTOUTm is toggled at the beginning of operation. |

#### (2) TAUDnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.60 Contents of the TAUDnCMURm Register for Clock Divide Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Detection of falling edge<br>01: Detection of rising edge<br>10: Detection of rising and falling edges<br>11: Setting prohibited |

### (3) Channel output mode

Table 25.61 Control Bit Settings in Independent Channel Output Mode 1

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 0: Independent channel output |
| TAUDnTOC.TAUDnTOCm | 0: Operating mode 1 (Toggle mode if TAUDnTOM.TAUDnTOMm = 0) |
| TAUDnTOL.TAUDnTOLm | 0: The setting is disabled in toggle mode. (The value after reset.) |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set these bits to 0 |
| TAUDnTRC.TAUDnTRCm | |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

### (4) Simultaneous rewrite

Simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the clock divide function. Therefore, these registers should be set to 0.

Table 25.62 Simultaneous Rewrite Settings for Clock Divide Function

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

### 25.12.3.5 Operating Procedure for Clock Divide Function

Table 25.63 Operating Procedure for Clock Divide Function

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers as described in **Table 25.59, Contents of the TAUDnCMORm Register for Clock Divide Function**, and **Table 25.60, Contents of the TAUDnCMURm Register for Clock Divide Function**.<br><br>Set the value of TAUDnCDRm register.<br><br>Set channel output mode by setting the control bits as described in **Table 25.61, Control Bit Settings in Independent Channel Output Mode 1**. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is set to 1 and the counter starts.<br>TAUDnCNTm loads TAUDnCDRm value. If TAUDnCMORm.TAUDnMD0 is set to 1, INTTAUDnIm is generated and TAUDTTOUTm is toggled. |
| During Operation | The value of TAUDnCDRm is changeable at any time.<br>The TAUDnCNTm register can be read at all times. | TAUDnCNTm counts down each time TAUDTTINm input edge is detected. When the counter reaches $0000_H$:<br>• TAUDnCDRm value is loaded in TAUDnCNTm and count operation continues.<br>• INTTAUDnIm is generated.<br>• TAUDTTOUTm is toggled.<br>Afterwards, this procedure is repeated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops. TAUDnCNTm and TAUDTTOUTm retain their current values. |

Restart Operation

### 25.12.3.6 Specific Timing Diagrams

#### (1) TAUDnCDRm = $0000_H$

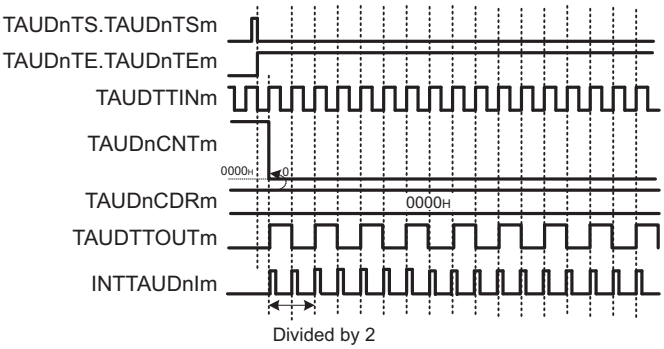



**Figure 25.40 TAUDnCDRm = $0000_H$, TAUDnCMORm.TAUDnMD0 = 1, TAUDnCMURm.TAUDnTIS[1:0] = $01_B$**

- If TAUDnCDRm is $0000_H$, TAUDnCNTm is always $0000_H$.
- INTTAUDnIm is generated every count clock, resulting in TAUDTTOUTm toggling every count clock.

**Figure 25.40** shows an operation timing example. Actually, there is a delay from TINm detection until TOUTm output because of the delay time of a noise filter or synchronization circuit placed between the TAUDnIm pin and TAUDn.

#### (2) Operation restart

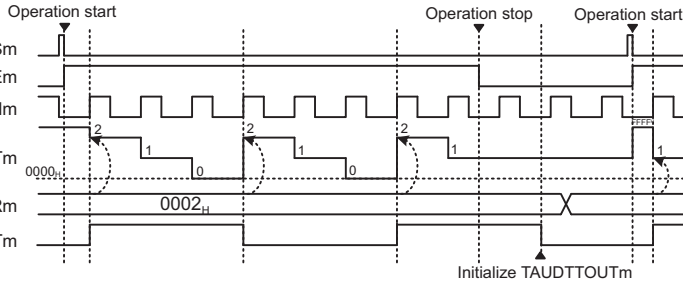



**Figure 25.41 Operation Restart (TAUDnCMORm.TAUDnMD0 = 1, TAUDnCMURm.TAUDnTIS[1:0] = $01_B$)**

To reset the value of TAUDTTOUTm:

- Set TAUDnTOE.TAUDnTOEm = 0 when the counter is stopped (TAUDnTE.TAUDnTEm = 0).
- Then, write either 0 or 1 to TAUDnTO.TAUDnTOm to set the new start value of TAUDTTOUTm.

### (3) Forced restart

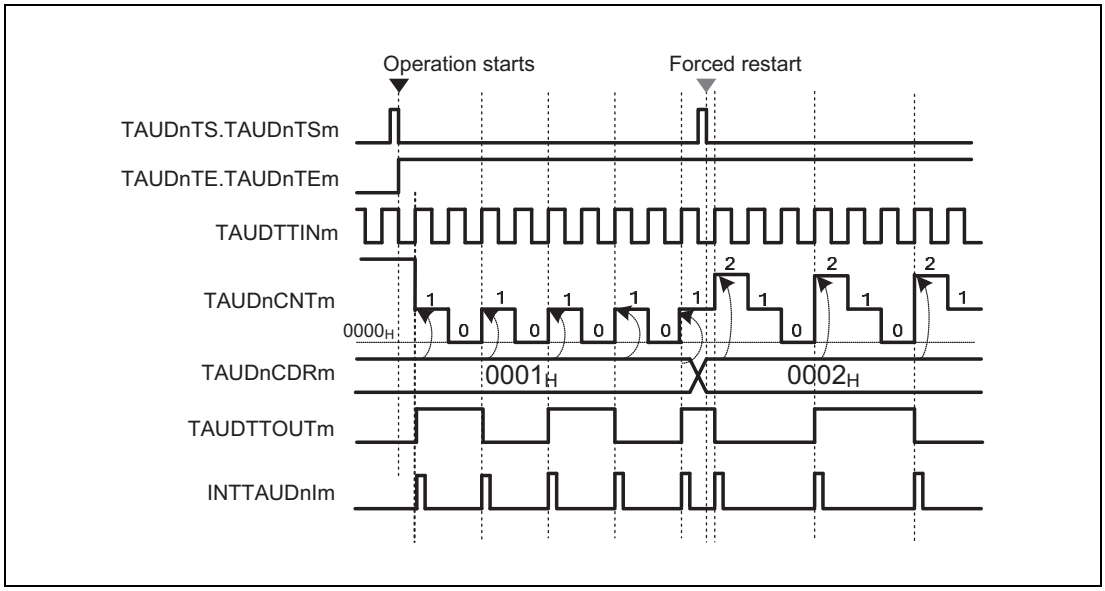



**Figure 25.42 Forced Restart Operation (TAUDnCMORm.TAUDnMD0 = 1, TAUDnCMURm.TAUDnTIS[1:0] = $01_B$)**

- The counter can be forcibly restarted (without stopping it first) by setting TAUDnTS.TAUDnTSm = 1 during operation.
- The value of TAUDnCDRm is written to TAUDnCNTm and the count operation restarts.
- TAUDTTOUTm restarts at the same level as before the forced restart.

## 25.12.4 External Event Count Function

### 25.12.4.1 Overview

#### Summary

This function is used as an event timer, which generates an interrupt (INTTAUDnIm) when a specific number of valid TAUDTTINm input edges are detected.

#### Prerequisites

- The operating mode should be set to the event count mode. (See **Table 25.64, Contents of the TAUDnCMORm Register for External Event Count Function**.)
- TAUDTTOUTm is not used with this function.

#### Functional description

The counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This in turn sets TAUDnTE.TAUDnTEm = 1, enabling count operation. When the counter starts, the current value of TAUDnCDRm is loaded into TAUDnCNTm.

When a valid TAUDTTINm input edge is detected, the value of TAUDnCNTm decrements by 1. TAUDnCNTm retains this value until a valid TAUDTTINm input edge is detected or the counter is restarted.

When the valid edge is detected for the (TAUDnCDRm + 1) times, INTTAUDnIm is generated. Then, TAUDnCDRm value is loaded into TAUDnCNTm to continue operation subsequently.

The counter can be stopped by setting TAUDnTT.TAUDnTTm to 1. This sets TAUDnTE.TAUDnTEm to 0. The counter can be restarted by setting TAUDnTS.TAUDnTSm to 1. The counter can also be restarted without stopping it first (forced restart) by setting TAUDnTS.TAUDnTSm to 1 during operation.

The value of TAUDnCDRm can be rewritten at any time, and the changed value of TAUDnCDRm is applied the next time the counter starts to count down.

#### Conditions

An edge type used as a trigger is specified by TAUDnCMURm.TAUDnTIS[1:0] bits.

- When TAUDnCMURm.TAUDnTIS[1:0] = $00_B$, falling edges are counted.
- When TAUDnCMURm.TAUDnTIS[1:0] = $01_B$, rising edges are counted.
- When TAUDnCMURm.TAUDnTIS[1:0] = $10_B$, both edges are counted.

### 25.12.4.2 Equations

Number of valid edges detected before INTTAUDnIm generation = TAUDnCDRm + 1

### 25.12.4.3 Block Diagram and General Timing Diagram

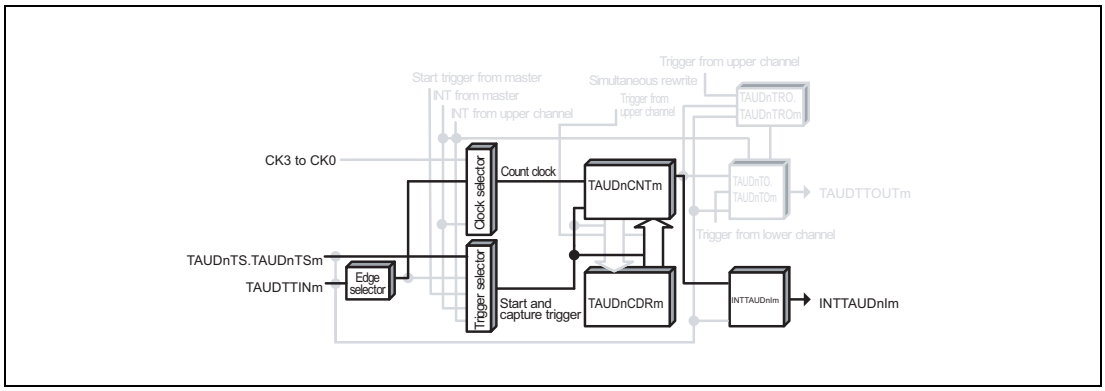



**Figure 25.43 Block Diagram of External Event Count Function**

The following settings apply to the general timing diagram.

- Detection of rising edge (TAUDnCMURm.TAUDnTIS[1:0] = $01_B$)

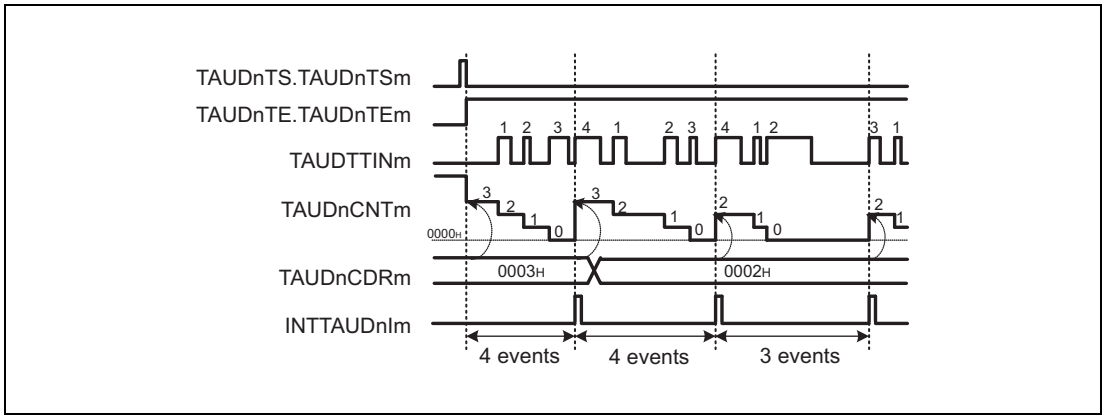



**Figure 25.44 General Timing Diagram of External Event Count Function**

### 25.12.4.4 Register Settings

#### (1) TAUDnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS [1:0] | | TAUDnCCS [1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS [1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.64 Contents of the TAUDnCMORm Register for External Event Count Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 01: Valid TAUDTTINm input edge is used as a count clock. |
| 11 | TAUDnMAS | 0: Independent operation. Set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 000: Trigger the counter using software. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0011: Event count mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm not generated at the beginning of operation. |

#### (2) TAUDnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.65 Contents of the TAUDnCMURm Register for External Event Count Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Falling edge is detected.<br>01: Rising edge is detected.<br>10: Both edges are detected.<br>11: Setting prohibited |

#### (3) Channel output mode

The channel output mode is not used by this function.

#### (4) Simultaneous rewrite

Simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the external event count function. Therefore, these registers should be set to 0.

**Table 25.66 Simultaneous Rewrite Settings for External Event Count Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

### 25.12.4.5 Operating Procedure for External Event Count Function

**Table 25.67 Operating Procedure for External Event Count Function**

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers as described in **Table 25.64, Contents of the TAUDnCMORm Register for External Event Count Function**, and **Table 25.65, Contents of the TAUDnCMURm Register for External Event Count Function**.<br><br>Set the value of TAUDnCDRm register. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is set to 1 and the counter starts.<br>TAUDnCNTm loads TAUDnCDRm value and waits for TAUDTTINm input edge detection. |
| During Operation | Detection of TAUDTTINm edge<br><br>The value of TAUDnCDRm is changeable at any time.<br><br>The TAUDnCNTm register can be read at any time. | TAUDnCNTm counts down each time TAUDTTINm input edge is detected. When effective edges are detected (TAUDnCDRm + 1) times:<br>• TAUDnCDRm value is loaded in TAUDnCNTm and count operation continues.<br>• INTTAUDnIm is generated.<br>Afterwards, this procedure is repeated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops and retains its current value. |

Restart Operation

### 25.12.4.6 Specific Timing Diagrams

#### (1) TAUDnCDRm = $0000_H$

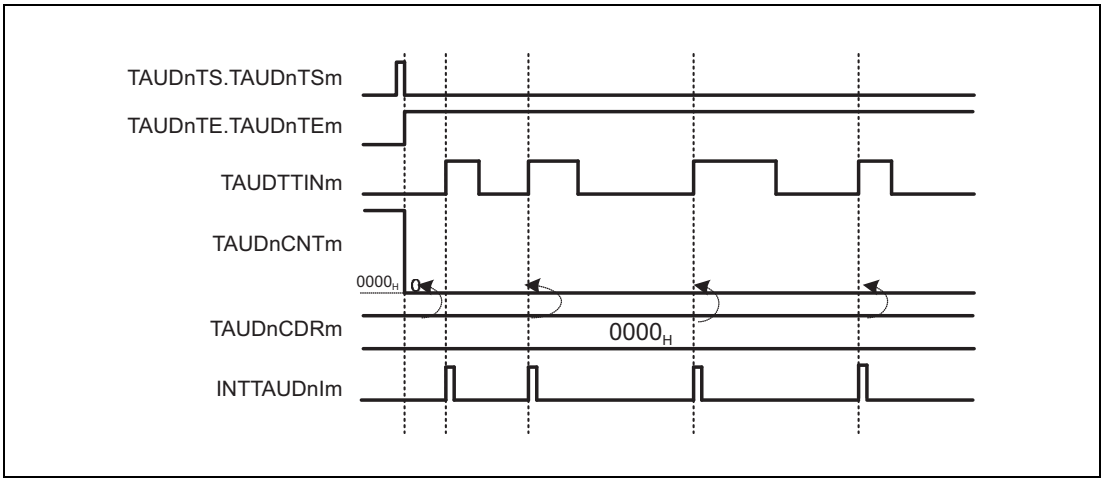



Figure 25.45 TAUDnCDRm = $0000_H$, TAUDnCMURm.TAUDnTIS[1:0] = $01_B$

- If $0000_H$ = TAUDnCDRm, $0000_H$ is loaded into TAUDnCNTm each time a valid TAUDTTINm input edge is detected.
  In other words, INTTAUDnIm is generated each time a valid TAUDTTINm input edge is detected.

#### (2) Operation stop and restart

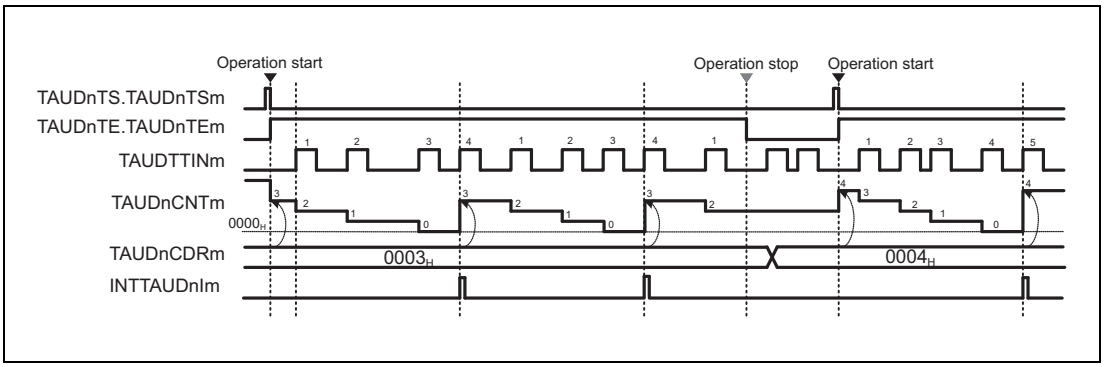



Figure 25.46 Operation Stop and Restart (TAUDnCMURm.TAUDnTIS[1:0] = $01_B$)

- The counter can be stopped by setting TAUDnTT.TAUDnTTm to 1. This sets TAUDnTE.TAUDnTEm to 0.
- TAUDnCNTm stops and retains its current value. TAUDTTINm continues and TAUDnCNTm ignores the valid edge.
- The counter can be restarted by setting TAUDnTS.TAUDnTSm to 1. TAUDnCNTm loads the TAUDnCDRm value and restarts count operation.

### (3) Forced restart

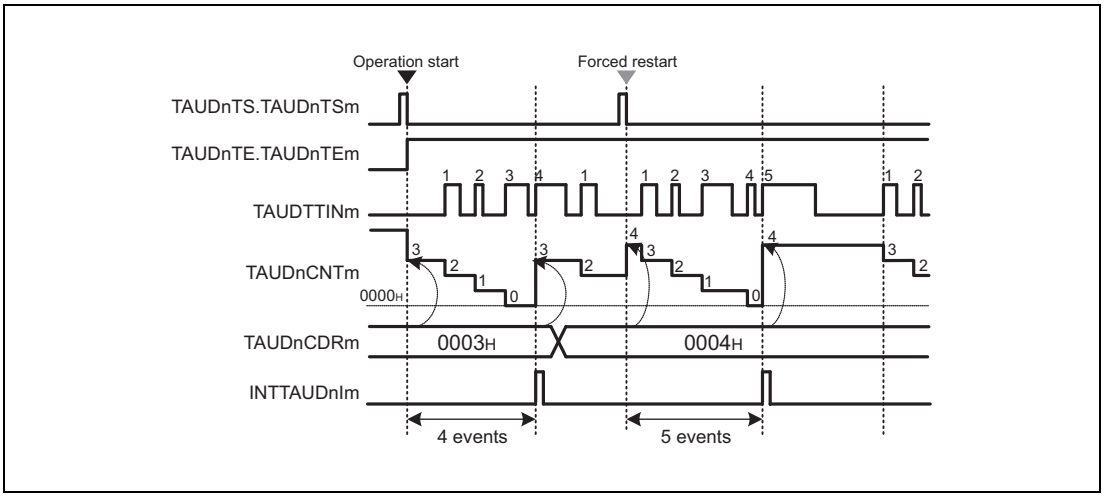



**Figure 25.47 Forced Restart Operation (TAUDnCMURm.TAUDnTIS[1:0] = $01_B$)**

Once a forced restart is made, the changed TAUDnCDRm value is applied to TAUDnCNTm immediately.

- The counter can be restarted without making a stop by setting TAUDnTS.TAUDnTSm to 1 during operation.
- The value of TAUDnCDRm is loaded into TAUDnCNTm and the counter awaits the next valid TAUDTTINm input edge.

## 25.12.5 Delay Count Function

### 25.12.5.1 Overview

#### Summary

This function generates interrupts (INTTAUDnIm), which have a defined delay to the TAUDTTINm input signal. TAUDTTINm input signal pulses that occur within the delay period are ignored.

#### Prerequisites

- The operating mode should be set to one-count mode. See **Table 25.68, Contents of the TAUDnCMORm Register for Delay Count Function**.
- TAUDTTOUTm is not used with this function.
- Trigger detection should be disabled during counting (TAUDnCMORn.TAUDnMD0 = 0).

#### Functional description

The counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This sets TAUDnTE.TAUDnTEm = 1, enabling count operation.

The counter starts when a valid TAUDTTINm input start edge is detected. The value of TAUDnCDRm is loaded into TAUDnCNTm and the counter starts to count down from the TAUDnCDRm value.

When the counter reaches $0000_H$, an interrupt is generated. The counter returns to $FFFF_H$ and awaits the next valid TAUDTTINm input edge.

When the counter is counting down, further TAUDTTINm input signals are ignored, i.e., the counter does not reset.

The value of TAUDnCDRm can be rewritten at any time, and the changed value of TAUDnCDRm is applied the next time the counter starts to count down.

#### Conditions

The type of edge used as a trigger is specified by the TAUDnCMURm.TAUDnTIS[1:0] bits.

- If TAUDnCMURm.TAUDnTIS[1:0] = $00_B$, falling edges trigger the counter.
- If TAUDnCMURm.TAUDnTIS[1:0] = $01_B$, rising edges trigger the counter.
- If TAUDnCMURm.TAUDnTIS[1:0] = $10_B$, rising and falling edges trigger the counter.

### 25.12.5.2 Equations

Delay between TAUDTTINm and INTTAUDnIm = count clock cycle × (TAUDnCDRm + 1)

### 25.12.5.3 Block Diagram and General Timing Diagram

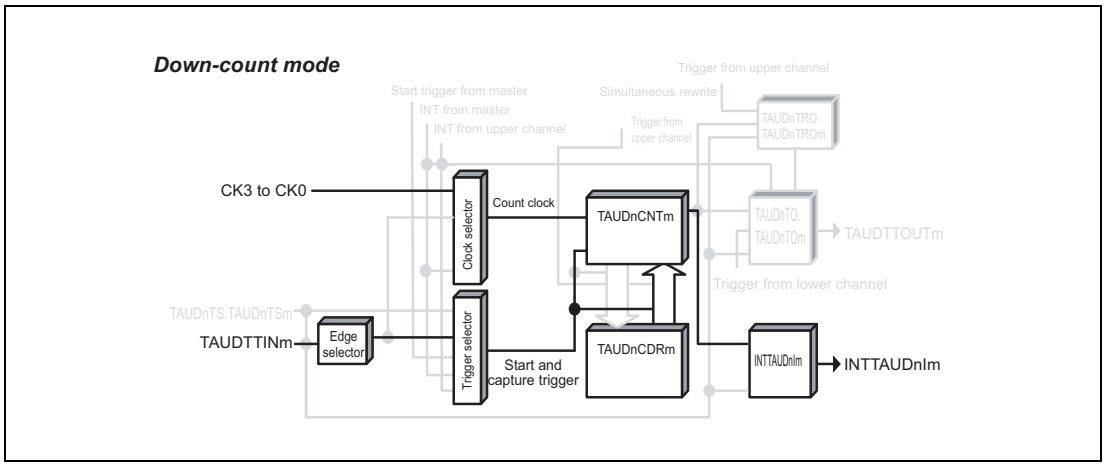



**Figure 25.48 Block Diagram of Delay Count Function**

The following settings apply to the general timing diagram.

- Detection of falling edge (TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)

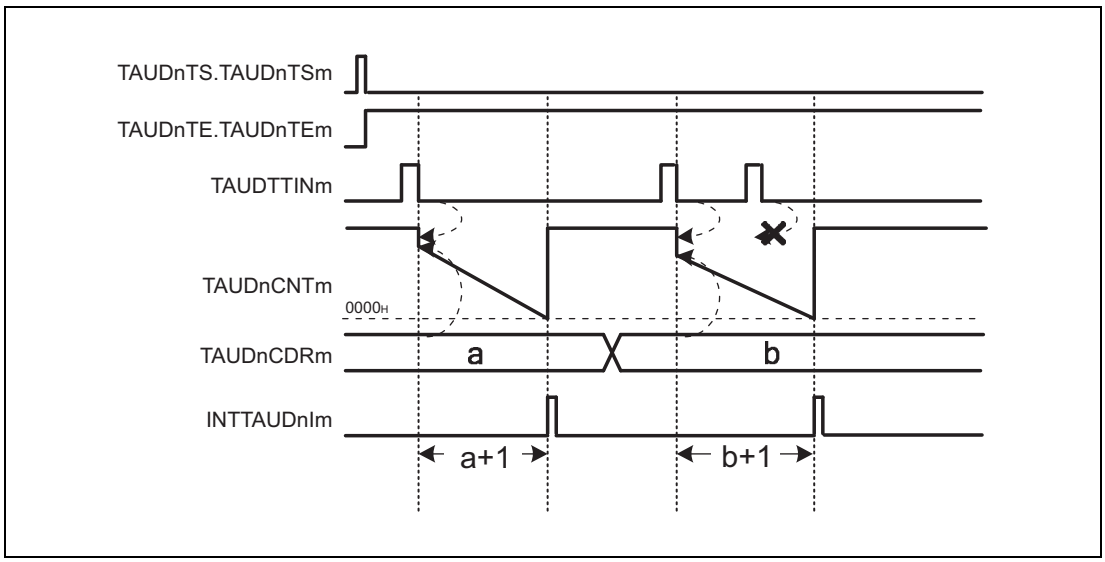



**Figure 25.49 General Timing Diagram of Delay Count Function**

### 25.12.5.4 Register Settings

#### (1) TAUDnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.68 Contents of the TAUDnCMORm Register for Delay Count Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation. Set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 001: Valid TAUDTTINm input edge signal is used as an external start trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0100: One-count mode |
| 0 | TAUDnMD0 | 0: Disables a start trigger during operation |

#### (2) TAUDnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.69 Contents of the TAUDnCMURm Register for Delay Count Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Detection of falling edge<br>01: Detection of rising edge<br>10: Detection of rising and falling edges<br>11: Setting prohibited |

#### (3) Channel output mode

TAUDnTOE.TAUDnTOEm is set to 0 because the channel output mode is not used with this function. However, this mode can be used in independent channel output mode controlled by software.

#### (4) Simultaneous rewrite

Simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the delay count function. Therefore, these registers should be set to 0.

**Table 25.70 Simultaneous Rewrite Settings for Delay Count Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

### 25.12.5.5 Operating Procedure for Delay Count Function

**Table 25.71 Operating Procedure for Delay Count Function**

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers as described in **Table 25.68, Contents of the TAUDnCMORm Register for Delay Count Function**, and **Table 25.69, Contents of the TAUDnCMURm Register for Delay Count Function**.<br><br>Set the value of TAUDnCDRm register. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0.<br><br>Detection of TAUDTTINm start edge | TAUDnTE.TAUDnTEm is set to 1 and TAUDnCNTm waits for detection of the TAUDTTINm start edge.<br><br>When a start edge is detected, the TAUDnCDRm value is loaded in TAUDnCNTm. |
| During Operation | The TAUDnCDRm register value can be changed at any time.<br>The TAUDnCNTm register can be read at all times. | TAUDnCNTm counts down. When the counter reaches $0000_H$.<br>INTTAUDnIm is generated.<br>TAUDnCNTm stops counting, returns $FFFF_H$, and waits for a trigger.<br><br>If a trigger occurs while TAUDnCNTm is counting, the trigger is ignored.<br>Afterwards, this procedure is repeated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops and retains its value. |

Restart Operation

## 25.12.6 One-Pulse Output Function

### 25.12.6.1 Overview

#### Summary

This function generates an interrupt (INTTAUDnIm) when a valid TAUDTTINm input edge is detected and at a defined interval afterward. TAUDTTINm input signal pulses that occur within the defined interval are ignored. When an interrupt is generated, the TAUDTTOUTm signal toggles, resulting in a square wave.

#### Prerequisites

- The operation mode should be set to pulse one-count mode. (See **Table 25.72, Contents of the TAUDnCMORm Register for One-Pulse Output Function**.)
- The channel output mode should be set to independent channel output mode 2. (See **Section 25.7, Channel Output Modes**.)
- Trigger detection should be disabled during counting (TAUDnCMORn.TAUDnMD0 = 0).

#### Functional description

The counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This in turn sets TAUDnTE.TAUDnTEm = 1, enabling count operation.

The counter starts when a valid TAUDTTINm input edge is detected. The value of TAUDnCDRm is loaded into TAUDnCNTm and the counter starts to count down from the TAUDnCDRm value. An interrupt is generated and TAUDTTOUTm is set to active level.

When the counter reaches $0001_H$, an interrupt is generated and TAUDTTOUTm is set to the inactive level. The counter stops at $0000_H$ and awaits the next valid TAUDTTINm input edge.

When the counter is counting down, further TAUDTTINm input signals are ignored, i.e., the counter does not reset.

The value of TAUDnCDRm can be rewritten at any time, and the changed value of TAUDnCDRm is applied the next time the counter starts to count down.

#### Conditions

The type of edge used as a trigger is specified by the TAUDnCMURm.TAUDnTIS[1:0] bits.

- If TAUDnCMURm.TAUDnTIS[1:0] = $00_B$, falling edges trigger the counter.
- If TAUDnCMURm.TAUDnTIS[1:0] = $01_B$, rising edges trigger the counter.
- If TAUDnCMURm.TAUDnTIS[1:0] = $10_B$, rising and falling edges trigger the counter.

### 25.12.6.2 Equations

Interval between TAUDTTINm and INTTAUDnIm = TAUDTTOUTm (timer output) width = count clock cycle × TAUDnCDRm

### 25.12.6.3 Block Diagram and General Timing Diagram

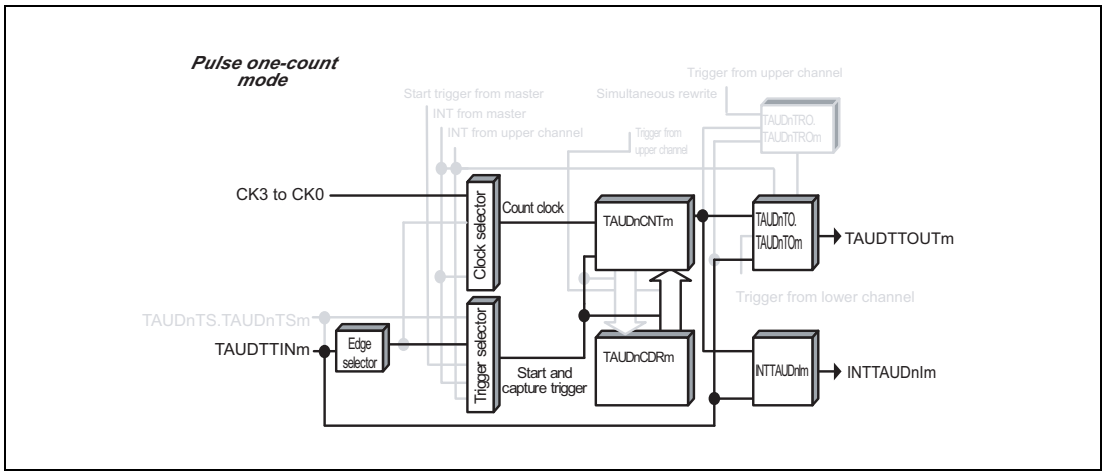



**Figure 25.50 Block Diagram of One-Pulse Output Function**

The following settings apply to the general timing diagram.

- Detection of falling edge (TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)

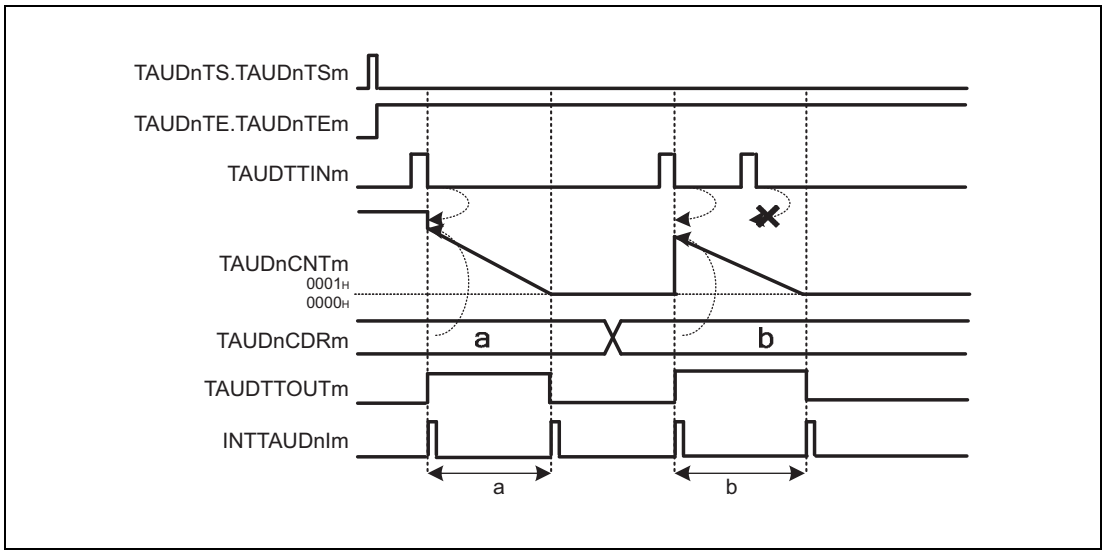



**Figure 25.51 General Timing Diagram of One-Pulse Output Function**

### 25.12.6.4 Register Settings

#### (1) TAUDnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.72 Contents of the TAUDnCMORm Register for One-Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation. Set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 001: Valid TAUDTTINm input edge signal is used as an external start trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 1010: Pulse one-count mode |
| 0 | TAUDnMD0 | 0: Disables a start trigger during operation. |

#### (2) TAUDnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.73 Contents of the TAUDnCMURm Register for One-Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Detection of falling edge<br>01: Detection of rising edge<br>10: Detection of rising and falling edges<br>11: Setting prohibited |

### (3) Channel output mode

Table 25.74 Control Bit Settings in Independent Channel Output Mode 2

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode controlled by software. |
| TAUDnTOM.TAUDnTOMm | 0: Independent channel output |
| TAUDnTOC.TAUDnTOCm | 1: Operating mode 2 |
| TAUDnTOL.TAUDnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set these bits to 0 |
| TAUDnTRC.TAUDnTRCm | |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

**NOTE**

The channel output mode can also be set to channel output mode controlled by software by setting TAUDnTOE.TAUDnTOEm = 0. TAUDTTOUTm can then be controlled independently of the interrupts. For details, see **Table 25.48, Channel Output Modes**.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the One-Pulse Output Function. Therefore, these registers should be set to 0.

Table 25.75 Simultaneous Rewrite Settings for One-Pulse Output Function

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

### 25.12.6.5 Operating Procedure for One-Pulse Output Function

Table 25.76 Operating Procedure for One-Pulse Output Function

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers as described in **Table 25.72, Contents of the TAUDnCMORm Register for One-Pulse Output Function**, and **Table 25.73, Contents of the TAUDnCMURm Register for One-Pulse Output Function**.<br><br>Set the value of TAUDnCDRm register.<br><br>Set channel output mode by setting the control bits as described in **Table 25.74, Control Bit Settings in Independent Channel Output Mode 2**. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0.<br><br>Detection of TAUDTTINm start edge | TAUDnTE.TAUDnTEm is set to 1 and TAUDnCNTm waits for detection of the TAUDTTINm start edge.<br><br>When a start edge is detected, TAUDnCNTm loads the TAUDnCDRm value. |
| During Operation | The value of TAUDnCDRm is changeable at any time.<br>The TAUDnCNTm register can be read at all times. | INTTAUDnIm is generated when TAUDnCNTm starts and TAUDTTOUTm is set to its active level. TAUDnCNTm counts down. When the counter reaches $0001_H$:<br>• INTTAUDnIm is generated.<br>• TAUDTTOUTm is set to its inactive level.<br>TAUDnCNTm stops counting and waits for a trigger.<br><br>If a trigger occurs while TAUDnCNTm is counting, the trigger is ignored. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm and TAUDTTOUTm stop and retain their current values. |

Restart Operation

## 25.12.7 TAUDTTINm Input Pulse Interval Measurement Function

### 25.12.7.1 Overview

#### Summary

This function captures the count value and uses this value and the overflow bit TAUDnCSRm.TAUDnOVF to measure the interval of the TAUDTTINm input signals.

#### Prerequisites

- The operating mode should be set to capture mode. See **Table 25.78, Contents of the TAUDnCMORm Register for TAUDTTINm Input Pulse Interval Measurement Function**.
- TAUDTTOUTm is not used with this function.

#### Functional description

The counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This in turn sets TAUDnTE.TAUDnTEm = 1, enabling count operation. The counter TAUDnCNTm starts to count up from $0000_H$. When a valid TAUDTTINm edge is detected, the value of TAUDnCNTm is captured, transferred to TAUDnCDRm, and an interrupt INTTAUDnIm is generated. The counter resets to $0000_H$ and subsequently continues operation.

If the counter reaches $FFFF_H$ before a valid TAUDTTINm edge is detected, it overflows to $0000_H$. The counter is reset to $0000_H$ and subsequently continues operation. The values transferred to TAUDnCDRm and TAUDnCSRm.TAUDnOVF respectively depend on the values of bits TAUDnCMORm.TAUDnCOS[1:0].

**Table 25.77 Effects of Overflow**

| TAUDnCMORm. TAUDnCOS[1:0] | When Overflow Occurs | | When a Valid TAUDTTINm Input is Detected | |
|---|---|---|---|---|
| | TAUDnCDRm | TAUDnCSRm. TAUDnOVF | TAUDnCDRm, TAUDnCNTm | TAUDnCSRm. TAUDnOVF |
| 00 | Unchanged | 0 | TAUDnCNTm loaded into TAUDnCDRm | 1 |
| 01 | | 1 | | |
| 10 | Set to $FFFF_H$ | 0 | TAUDnCNTm set to 0, TAUDnCDRm unchanged | Unchanged |
| 11 | | 1 | | |

When TAUDnCMORm.TAUDnCOS[0] = 1, the overflow bit (TAUDnCSRm.TAUDnOVF) can be cleared only by setting TAUDnCSCm.TAUDnCLOV = 1.

The combination of the value of TAUDnCDRm and TAUDnCSRm.TAUDnOVF can be used to deduce the interval of the TAUDTTINm signal. However, if an overflow occurs multiple times before a valid TAUDTTINm input is detected, the overflow bit TAUDnCSRm.TAUDnOVF cannot indicate the occurrence of multiple overflows.

The function can be stopped by setting TAUDnTT.TAUDnTTm = 1. This sets TAUDnTE.TAUDnTEm = 0. TAUDnCNTm stops but retains its value. While the function is stopped, valid TAUDTTINm input edge detection and TAUDnCNTm capture are not performed.

The counter is reset to $0000_H$ and subsequently continues operation.

**Conditions**

If the TAUDnCMORm.TAUDnMD0 bit is set to 0, the first interrupt after a start or restart is not generated. For details, see **Section 25.9, TAUDTTOUTm Output and INTTAUDnIm Generation when Counter Starts or Restarts**.

**NOTE**

When TAUDnCMORm.TAUDnCOS[1:0] = $10_B$ or $11_B$, the value of TAUDnCNTm is not loaded into TAUDnCDRm when the first valid TAUDTTINm input edge occurs after an overflow. However, an interrupt is generated.

### 25.12.7.2 Equations

TAUDTTINm input pulse interval = count clock cycle × [(TAUDnCSRm.TAUDnOVF × ($FFFF_H$ + 1)) + TAUDnCDRm capture value + 1]

### 25.12.7.3 Block Diagram and General Timing Diagram

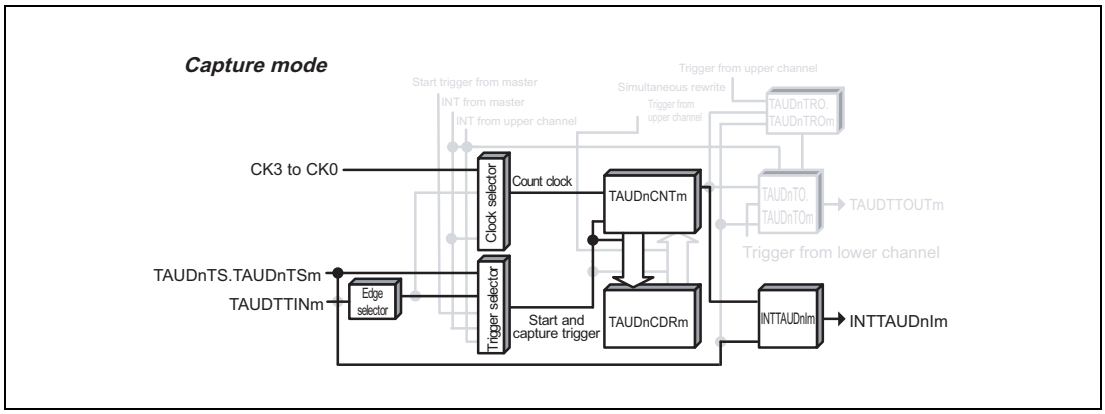



**Figure 25.52 Block Diagram of TAUDTTINm Input Pulse Interval Measurement Function**

The following settings apply to the general timing diagram.

- INTTAUDnIm is not generated at the beginning of operation (TAUDnCMORm.TAUDnMD0 = 0).
- Falling edge detection (TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)
- When a valid TAUDTTINm input is detected after an overflow, TAUDnCDRm is changed and TAUDnCSRm.TAUDnOVF is set to 1 (TAUDnCMORm.TAUDnCOS[1:0] = $00_B$).

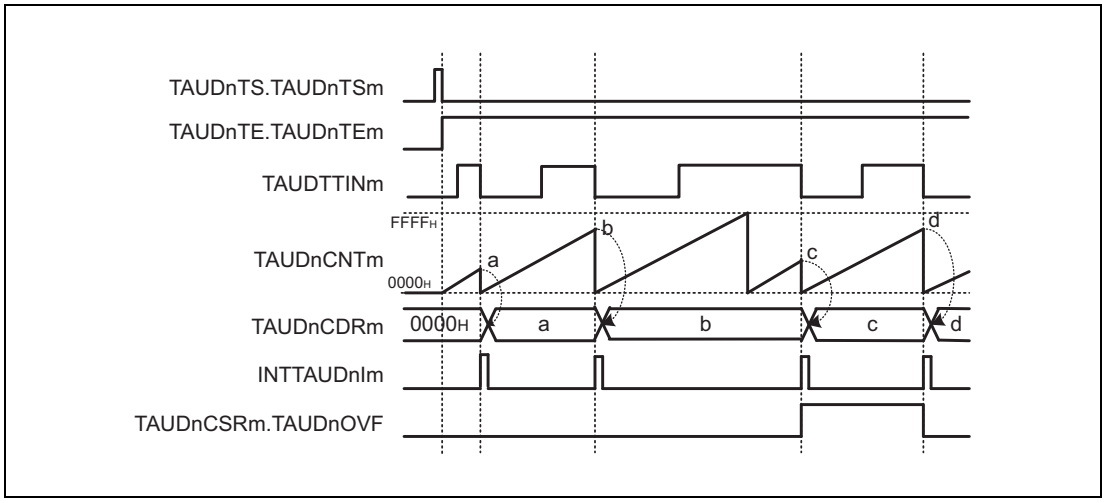



**Figure 25.53 General Timing Diagram of TAUDTTINm Input Pulse Interval Measurement Function**

### 25.12.7.4 Register Settings

#### (1) TAUDnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.78 Contents of the TAUDnCMORm Register for TAUDTTINm Input Pulse Interval Measurement Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation, set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 001: Valid edge of the TAUDTTINm input signal is used as the external capture trigger. |
| 7, 6 | TAUDnCOS[1:0] | See **Table 25.77, Effects of Overflow**. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0010: Capture mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm not generated at the beginning of operation.<br>1: INTTAUDnIm generated at the beginning of operation. |

#### (2) TAUDnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.79 Contents of the TAUDnCMURm Register for TAUDTTINm Input Pulse Interval Measurement Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Detection of falling edge<br>01: Detection of rising edge<br>10: Detection of rising and falling edges<br>11: Setting prohibited |

### (3) Channel output mode

TAUDnTOE.TAUDnTOEm is set to 0 because the channel output mode is not used with this function.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the TAUDTTINm input pulse interval measurement function. Therefore, these registers should be set to 0.

**Table 25.80 Simultaneous Rewrite Settings for TAUDTTINm Input Pulse Interval Measurement Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

## 25.12.7.5 Operating Procedure for TAUDTTINm Input Pulse Interval Measurement Function

**Table 25.81 Operating Procedure for TAUDTTINm Input Pulse Interval Measurement Function**

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers as described in **Table 25.78, Contents of the TAUDnCMORm Register for TAUDTTINm Input Pulse Interval Measurement Function**, and **Table 25.79, Contents of the TAUDnCMURm Register for TAUDTTINm Input Pulse Interval Measurement Function**.<br><br>The TAUDnCDRm register functions as a capture register. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is set to 1 and the counter starts.<br>TAUDnCNTm is cleared to $0000_H$. INTTAUDnIm is generated when TAUDnCMORm.TAUDnMD0 is set to 1. |
| During Operation | Detection of TAUDTTINm edge<br><br>The values of TAUDnCMURm.TAUDnTIS[1:0] bits can be changed at any time. The TAUDnCDRm and TAUDnCSRm registers can be read at any time.<br>TAUDnCSCm.TAUDnCLOV can be written to 1. (TAUDnCSRm.TAUDnOVF bit is cleared to 0.) | TAUDnCNTm starts to count up from $0000_H$.<br>When a TAUDTTINm valid edge is detected:<br>• TAUDnCNTm transfers (captures) its value to TAUDnCDRm, and returns to $0000_H$.<br>• INTTAUDnIm is then generated.<br>Afterwards, this procedure is repeated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops and both it and TAUDnCSRm.TAUDnOVF retain their current values. |

Restart Operation

### 25.12.7.6 Specific Timing Diagrams: Overflow Operation

#### (1) TAUDnCMORm.TAUDnCOS[1:0] = $00_B$

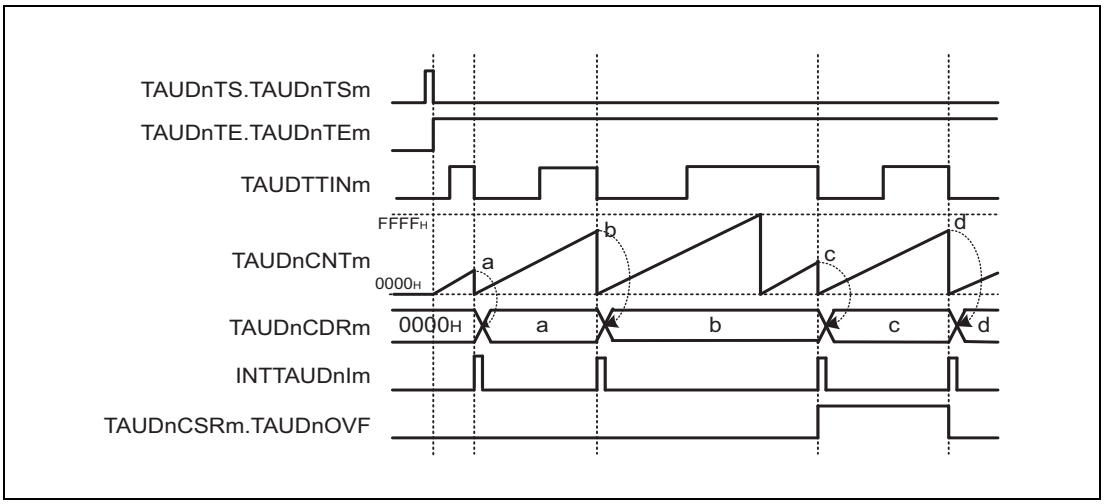



**Figure 25.54 TAUDnCMORm.TAUDnCOS[1:0] = $00_B$, TAUDnCMORm.TAUDnMD0 = 0, TAUDnCMURm.TAUDnTIS[1:0] = $00_B$**

- When an overflow occurs, the value of TAUDnCDRm remains unchanged and TAUDnCSRm.TAUDnOVF remains 0.
- Upon detection of the next valid TAUDTTINm input edge, the value of TAUDnCNTm is loaded into TAUDnCDRm and TAUDnCSRm.TAUDnOVF is set to 1.
- Upon detection of the next valid TAUDTTINm input edge with no overflow occurring, TAUDnCSRm.TAUDnOVF is cleared to 0.

#### (2) TAUDnCMORm.TAUDnCOS[1:0] = $01_B$

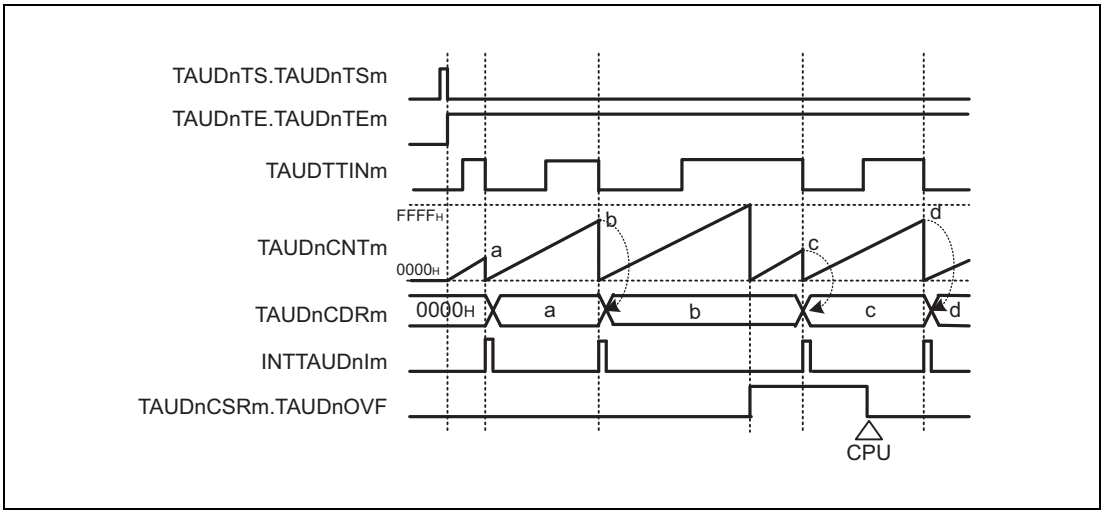



**Figure 25.55 TAUDnCMORm.TAUDnCOS[1:0] = $01_B$, TAUDnCMORm.TAUDnMD0 = 0, TAUDnCMURm.TAUDnTIS[1:0] = $00_B$**

- When an overflow occurs, the value of TAUDnCDRm remains unchanged and TAUDnCSRm.TAUDnOVF is set to1.

- Upon detection of the next valid TAUDTTINm input edge, the value of TAUDnCNTm is loaded into TAUDnCDRm.
- TAUDnCSRm.TAUDnOVF is only cleared by a CPU command (by setting TAUDnCSCm.TAUDnCLOV bit to 1).

### (3) TAUDnCMORm.TAUDnCOS[1:0] = $10_B$

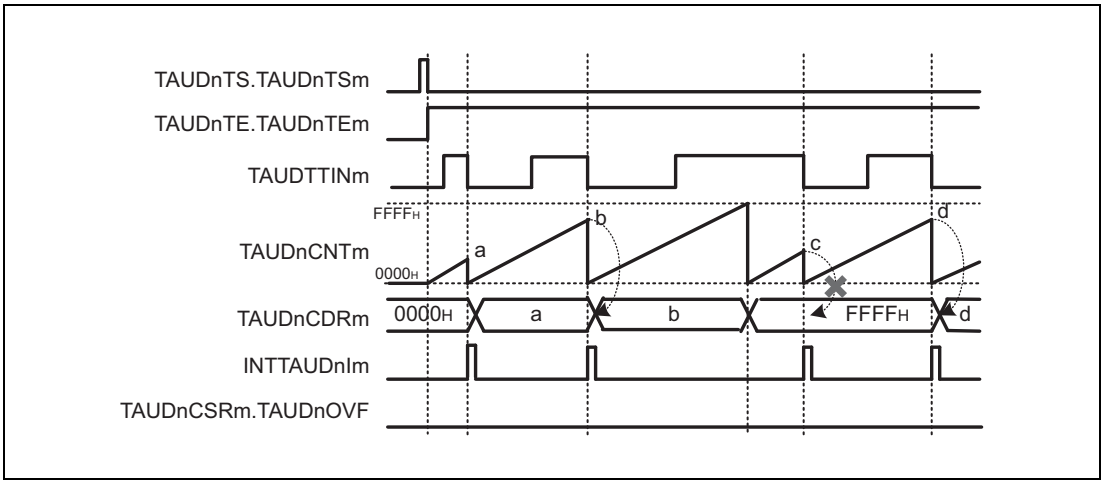



**Figure 25.56 TAUDnCMORm.TAUDnCOS[1:0] = $10_B$, TAUDnCMORm.TAUDnMD0 = 0, TAUDnCMURm.TAUDnTIS[1:0] = $00_B$**

- When an overflow occurs, TAUDnCDRm is set to $FFFF_H$ and TAUDnCSRm.TAUDnOVF remains 0.
- Upon detection of the next valid TAUDTTINm input edge, TAUDnCNTm is reset to 0, but TAUDnCDRm and TAUDnCSRm.TAUDnOVF remain unchanged.
- Thus, the next valid TAUDTTINm input edge after the overflow is ignored.

### (4) TAUDnCMORm.TAUDnCOS[1:0] = $11_B$

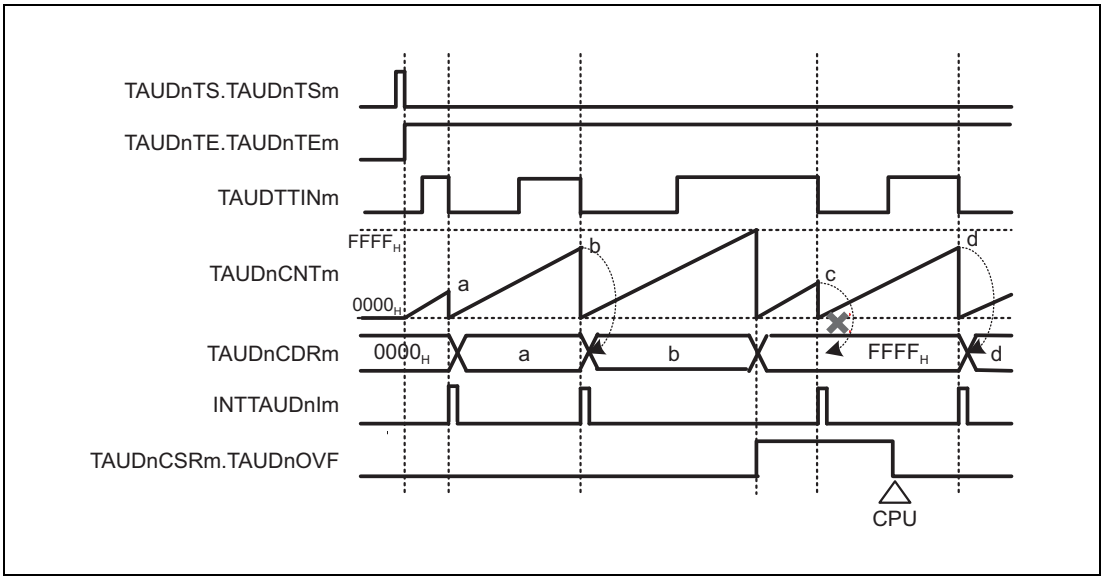



**Figure 25.57 TAUDnCMORm.TAUDnCOS[1:0] = $11_B$, TAUDnCMORm.TAUDnMD0 = 0, TAUDnCMURm.TAUDnTIS[1:0] = $00_B$**

- When an overflow occurs, TAUDnCDRm is set to $FFFF_H$ and TAUDnCSRm.TAUDnOVF is set to 1.
- Upon detection of the next valid TAUDTTINm input edge, TAUDnCNTm is reset to 0, but TAUDnCDRm and TAUDnCSRm.TAUDnOVF remain unchanged.
- Thus, the next valid TAUDTTINm input edge after the overflow is ignored.
- TAUDnCSRm.TAUDnOVF is cleared by setting TAUDnCSCm.TAUDnCLOV to 1.

## 25.12.8 TAUDTTINm Input Signal Width Measurement Function

### 25.12.8.1 Overview

#### Summary

This function measures the width of a TAUDTTINm signal, by starting the count at one edge of TAUDTTINm and capturing the count value at the other edge.

#### Prerequisites

- The operating mode should be set to capture and one-count mode. See **Table 25.83, Contents of the TAUDnCMORm Register for TAUDTTINm Input Signal Width Measurement Function**.
- TAUDTTOUTm is not used with this function.
- TAUDnCMORm.TAUDnMD0 should be set to 0.

#### Functional description

The counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This in turn sets TAUDnTE.TAUDnTEm = 1, enabling count operation. When a valid TAUDTTINm start edge is detected, the counter TAUDnCNTm starts to count up from $0000_H$. When a valid TAUDTTINm stop edge is detected, the value of TAUDnCNTm is captured, transferred to TAUDnCDRm, and an interrupt INTTAUDnIm is generated. The counter retains its value (TAUDnCDRm +1) and awaits the next valid TAUDTTINm input start edge.

If the counter reaches $FFFF_H$ before a valid TAUDTTINm stop edge is detected, it overflows. The counter is reset to $0000_H$ and subsequently continues operation. The values transferred to TAUDnCDRm and TAUDnCSRm.TAUDnOVF respectively depend on the values of bits TAUDnCMORm.TAUDnCOS[1:0].

**Table 25.82 Effects of Overflow**

| TAUDnCMORm.TAUDnCOS[1:0] | When Overflow Occurs | | When a Valid TAUDTTINm Input Stop Edge is Detected | |
|---|---|---|---|---|
| | TAUDnCDRm | TAUDnCSRm.TAUDnOVF | TAUDnCDRm, TAUDnCNTm | TAUDnCSRm.TAUDnOVF |
| 00 | Unchanged | 0 | TAUDnCNTm loaded into TAUDnCDRm | 1 |
| 01 | | 1 | | |
| 10 | Set to $FFFF_H$ | 0 | TAUDnCNTm stops counting<br>TAUDnCDRm unchanged | Unchanged |
| 11 | | 1 | | |

When TAUDnCMORm.TAUDnCOS[0] = 1, overflow bit TAUDnCSRm.TAUDnOVF can be cleared only by setting TAUDnCSCm.TAUDnCLOV to 1.

The combination of the value of TAUDnCDRm and TAUDnCSRm.TAUDnOVF can be used to deduce the width of the TAUDTTINm signal. However, if an overflow occurs multiple times before a valid TAUDTTINm input is detected, overflow bit TAUDnCSRm.TAUDnOVF cannot indicate the occurrence of multiple overflows.

This function cannot be forcibly restarted.

**NOTE**

When TAUDnCMORm.TAUDnCOS[1] = 1, the value of TAUDnCNTm is not loaded to TAUDnCDRm when the first valid TAUDTTINm input edge occurs after an overflow. However, an interrupt is generated.

### 25.12.8.2 Equations

TAUDTTINm input signal width = count clock cycle × [(TAUDnCSRm.TAUDnOVF × ($FFFF_H$ + 1)) + TAUDnCDRm capture value + 1]

### 25.12.8.3 Block Diagram and General Timing Diagram

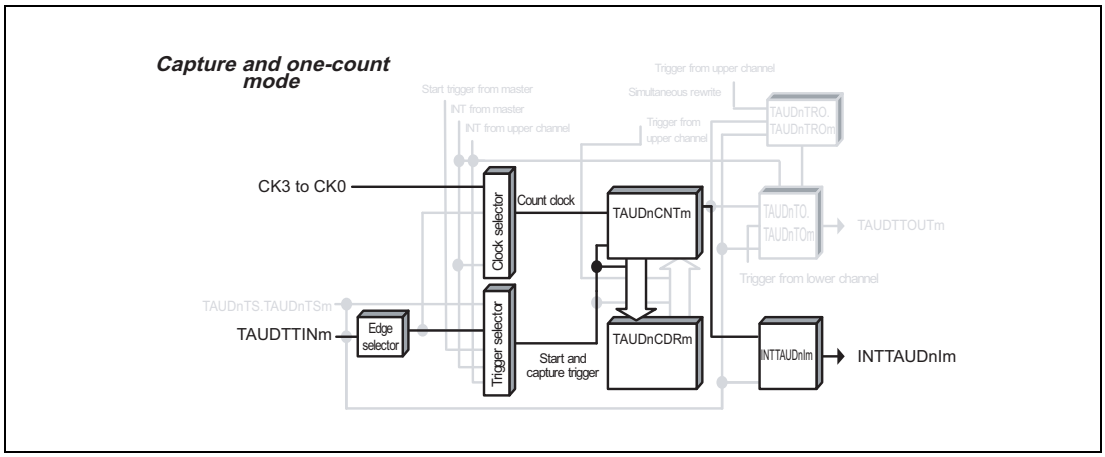



**Figure 25.58 Block Diagram of TAUDTTINm Input Signal Width Measurement Function**

The following settings apply to the general timing diagram.

- Detection of rising and falling edges = high width measurement (TAUDnCMURm.TAUDnTIS[1:0] = $11_B$)
- When a valid TAUDTTINm input is detected after an overflow, TAUDnCDRm is changed and TAUDnCSRm.TAUDnOVF is set to 1. (TAUDnCMORm.TAUDnCOS[1:0] = $00_B$)

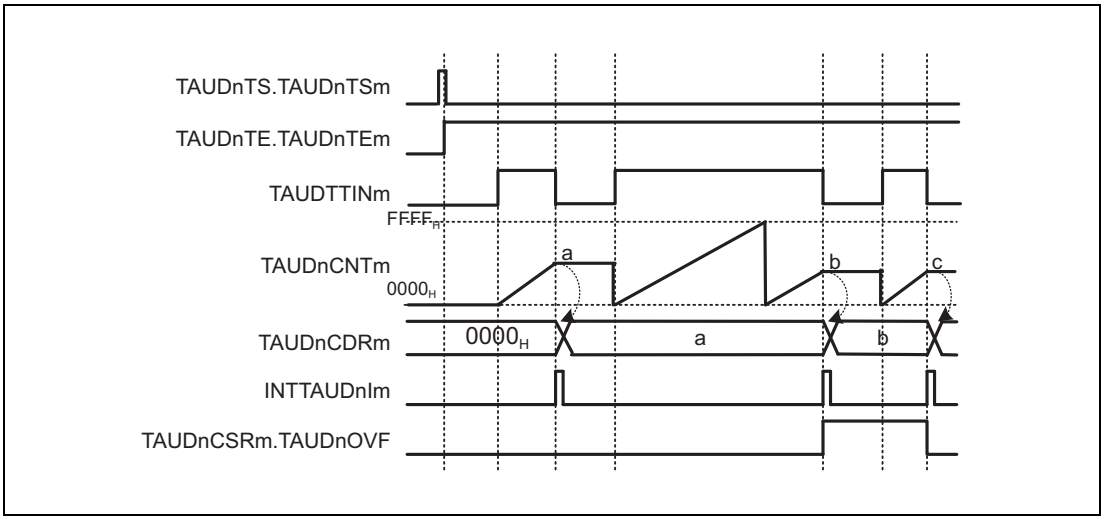



**Figure 25.59 General Timing Diagram of TAUDTTINm Input Signal Width Measurement Function**

### 25.12.8.4 Register Settings

#### (1) TAUDnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.83 Contents of the TAUDnCMORm Register for TAUDTTINm Input Signal Width Measurement Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation, set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 010: Valid edge of the TAUDTTINm input signal is used as an external start trigger and the reverse edge as a stop trigger. |
| 7, 6 | TAUDnCOS[1:0] | See **Table 25.82, Effects of Overflow**. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0110: Capture and one-count mode |
| 0 | TAUDnMD0 | 0: Disables the start trigger during operation. |

#### (2) TAUDnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.84 Contents of the TAUDnCMURm Register For TAUDTTINm Input Signal Width Measurement Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 10: Detection of rising and falling edges (low width measurement)<br>11: Detection of rising and falling edges (high width measurement) |

### (3) Channel output mode

TAUDnTOE.TAUDnTOEm is set to 0 because the channel output mode is not used with this function.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the TAUDTTINm input signal width measurement function. Therefore, these registers should be set to 0.

**Table 25.85 Simultaneous Rewrite Settings for TAUDTTINm Input Signal Width Measurement Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

## 25.12.8.5 Operating Procedure for TAUDTTINm Input Signal Width Measurement Function

**Table 25.86 Operating Procedure for TAUDTTINm Input Signal Width Measurement Function**

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers as described in **Table 25.83, Contents of the TAUDnCMORm Register for TAUDTTINm Input Signal Width Measurement Function**, and **Table 25.84, Contents of the TAUDnCMURm Register For TAUDTTINm Input Signal Width Measurement Function**.<br><br>The TAUDnCDRm register functions as a capture register. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is set to 1 and TAUDnCNTm waits for detection of the TAUDTTINm start edge.<br>When a TAUDTTINm start edge is detected, TAUDnCNTm starts to count up. |
| During Operation | TAUDnCDRm, TAUDnCNTm, and TAUDnCSRm registers can be read at any time.<br>TAUDnCSCm.TAUDnCLOV bit can be set to 1. | TAUDnCNTm starts to count up from $0000_H$.<br>When TAUDTTINm valid edge is detected:<br>• TAUDnCNTm transfers (captures) its value to TAUDnCDRm, and retains its value. INTTAUDnIm is then generated.<br>Counting stops at the “value that transferred to TAUDnCDRm + 1” and TAUDnCNTm waits for detection of the TAUDTTINm start edge.<br>Afterwards, this procedure is repeated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops and both it and TAUDnCSRm.TAUDnOVF retain their current values. |

Restart Operation

### 25.12.8.6 Specific Timing Diagrams: Overflow Operation

#### (1) TAUDnCMORm.TAUDnCOS[1:0] = $00_B$

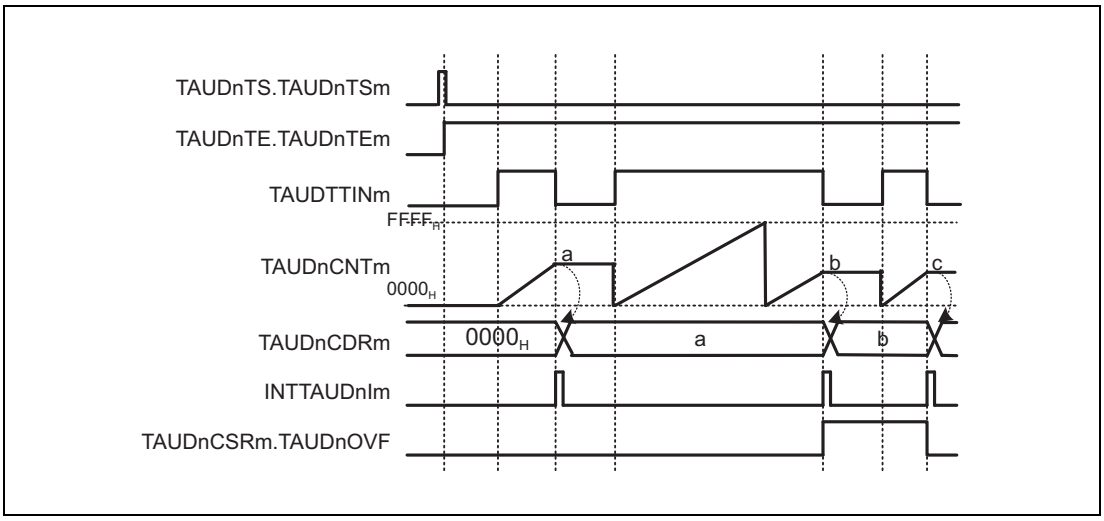



**Figure 25.60 TAUDnCMORm.TAUDnCOS[1:0] = $00_B$, TAUDnCMORm.TAUDnMD0 = 0, TAUDnCMURm.TAUDnTIS[1:0] = $11_B$**

- When an overflow occurs, the value of TAUDnCDRm remains unchanged and TAUDnCSRm.TAUDnOVF remains 0.
- Upon detection of the next valid TAUDTTINm input edge, the value of TAUDnCNTm is loaded into TAUDnCDRm and TAUDnCSRm.TAUDnOVF is set to 1.
- Upon detection of the next valid TAUDTTINm input edge with no overflow occurring, TAUDnCSRm.TAUDnOVF is cleared to 0.

### (2) TAUDnCMORm.TAUDnCOS[1:0] = $01_B$

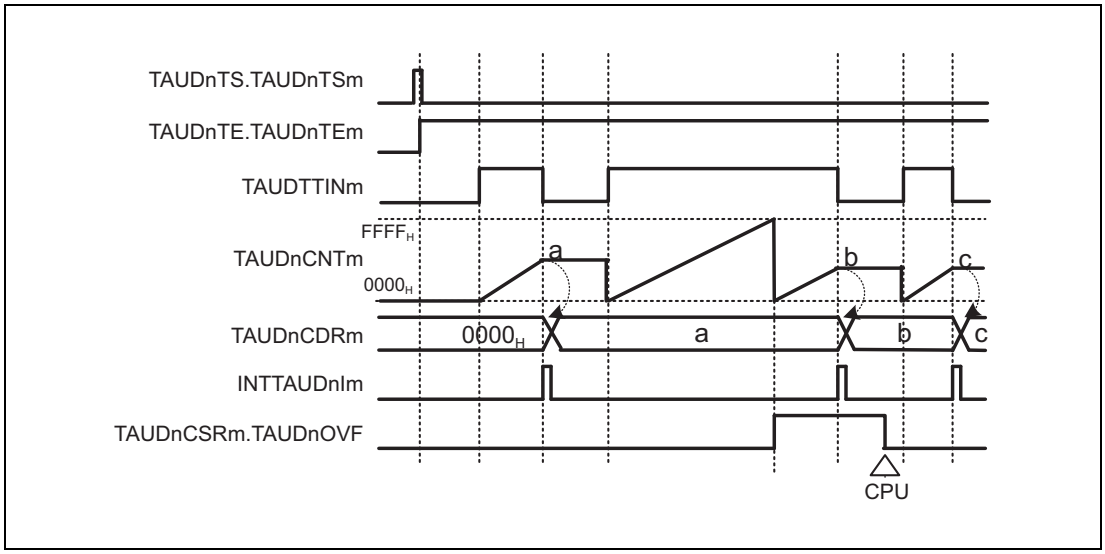



**Figure 25.61 TAUDnCMORm.TAUDnCOS[1:0] = $01_B$, TAUDnCMORm.TAUDnMD0 = 0, TAUDnCMURm.TAUDnTIS[1:0] = $11_B$**

- When an overflow occurs, the value of TAUDnCDRm remains unchanged and TAUDnCSRm.TAUDnOVF is set to 1.
- Upon detection of the next valid TAUDTTINm input edge, the value of TAUDnCNTm is loaded into TAUDnCDRm.
- TAUDnCSRm.TAUDnOVF is only cleared by a CPU command (by setting TAUDnCSCm.TAUDnCLOV bit to 1).

### (3) TAUDnCMORm.TAUDnCOS[1:0] = $10_B$

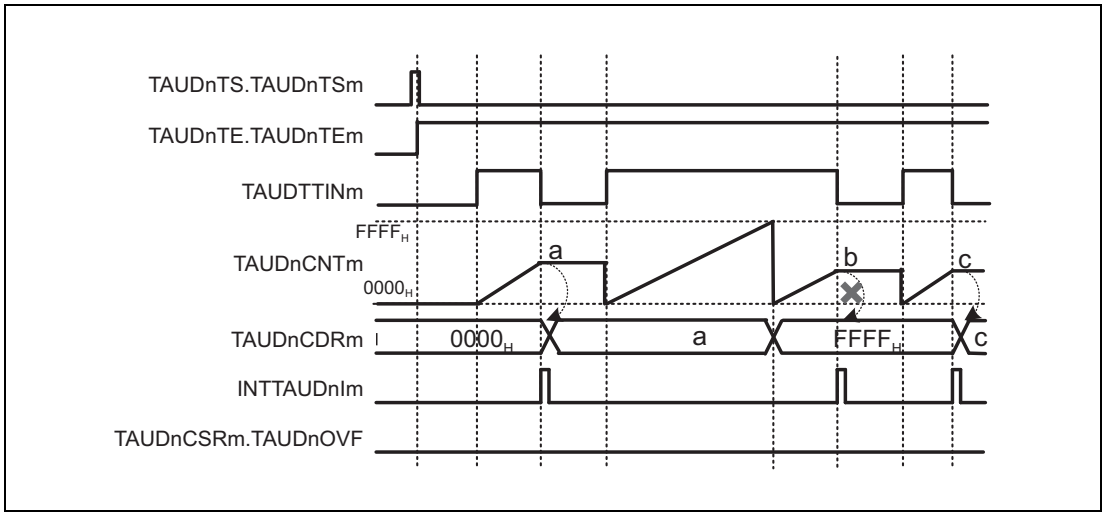



**Figure 25.62 TAUDnCMORm.TAUDnCOS[1:0] = $10_B$, TAUDnCMORm.TAUDnMD0 = 0, TAUDnCMURm.TAUDnTIS[1:0] = $11_B$**

- When an overflow occurs, TAUDnCDRm is set to $FFFF_H$ and TAUDnCSRm.TAUDnOVF remains 0.
- Upon detection of the next valid TAUDTTINm input edge, TAUDnCNTm stops counting, but TAUDnCDRm and TAUDnCSRm.TAUDnOVF remain unchanged.
- Thus, the next valid TAUDTTINm input edge after the overflow is ignored.

### (4) TAUDnCMORm.TAUDnCOS[1:0] = $11_B$

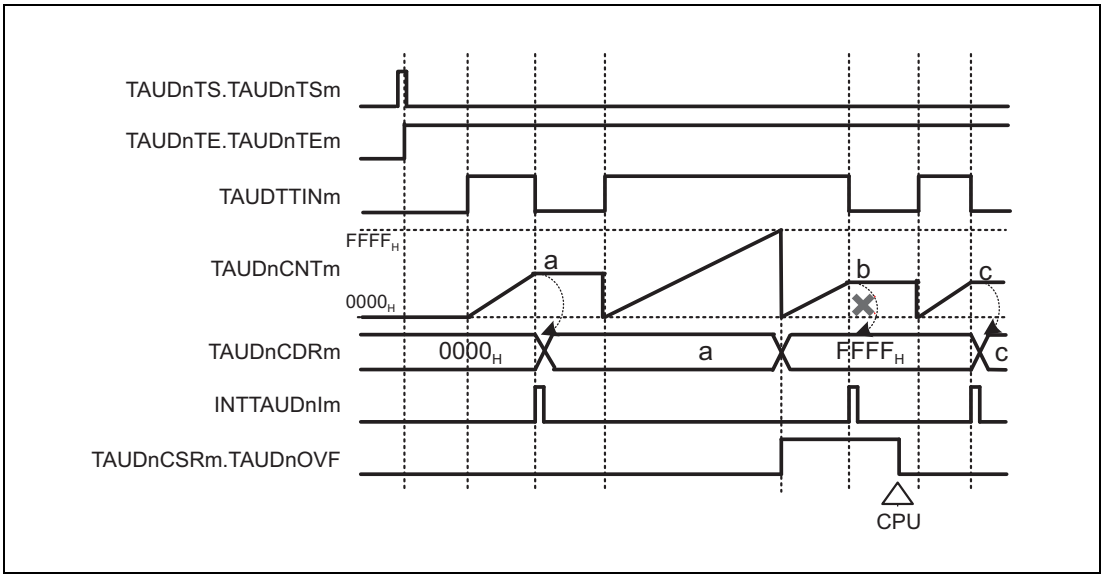



**Figure 25.63 TAUDnCMORm.TAUDnCOS[1:0] = $11_B$, TAUDnCMORm.TAUDnMD0 = 0, TAUDnCMURm.TAUDnTIS[1:0] = $11_B$**

- When an overflow occurs, TAUDnCDRm is set to $FFFF_H$ and TAUDnCSRm.TAUDnOVF is set to 1.
- Upon detection of the next valid TAUDTTINm input edge, TAUDnCNTm stops counting, but TAUDnCDRm and TAUDnCSRm.TAUDnOVF remain unchanged.
- Thus, the next valid TAUDTTINm input edge after the overflow is ignored.
- TAUDnCSRm.TAUDnOVF is cleared by setting TAUDnCSCm.TAUDnCLOV to 1.

## 25.12.9 TAUDTTINm Input Position Detection Function

### 25.12.9.1 Overview

#### Summary

This function measures the input signal duration by capturing the count value at the valid edge of TAUDTTINm.

#### Prerequisites

- The operating mode should be set to count capture mode. (See **Table 25.87, Contents of the TAUDnCMORm Register for TAUDTTINm Input Position Detection Function**.)
- TAUDTTOUTm is not used with this function.

#### Functional description

The counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This sets TAUDnTE.TAUDnTEm = 1, enabling count operation. The counter starts counting from $0000_H$. When a valid TAUDTTINm input edge is detected, the current value of TAUDnCNTm is loaded into TAUDnCDRm and an interrupt (INTTAUDnIm) is generated. The count operation continues.

When the counter reaches $FFFF_H$, the counter restarts from $0000_H$.

**NOTE**

The TAUDTTINm input signal is sampled at the frequency of the operation clock, specified by TAUDnCMORm.TAUDnCKS[1:0] bits. As a result, the output cycle of TAUDTTOUTm has an error of ± 1 operation clock cycle.

#### Conditions

If the TAUDnCMORm.TAUDnMD0 bit is set to 0, the first interrupt does not occur at the beginning of operation or after restart. For details, see **Section 25.9, TAUDTTOUTm Output and INTTAUDnIm Generation when Counter Starts or Restarts**.

### 25.12.9.2 Equations

Functional duration at a TAUDTTINm input pulse =

count clock cycle × (TAUDnCDRm capture value + 1)

### 25.12.9.3 Block Diagram and General Timing Diagram

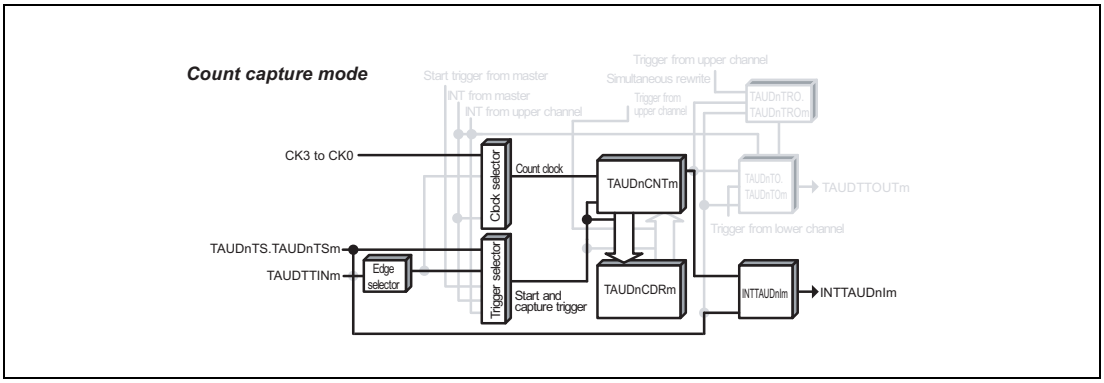



**Figure 25.64 Block Diagram of TAUDTTINm Input Position Detection Function**

The following settings apply to the general timing diagram.

- INTTAUDnIm is not generated at the beginning of operation. (TAUDnCMORm.TAUDnMD0 = 0)
- Detection of falling edge (TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)

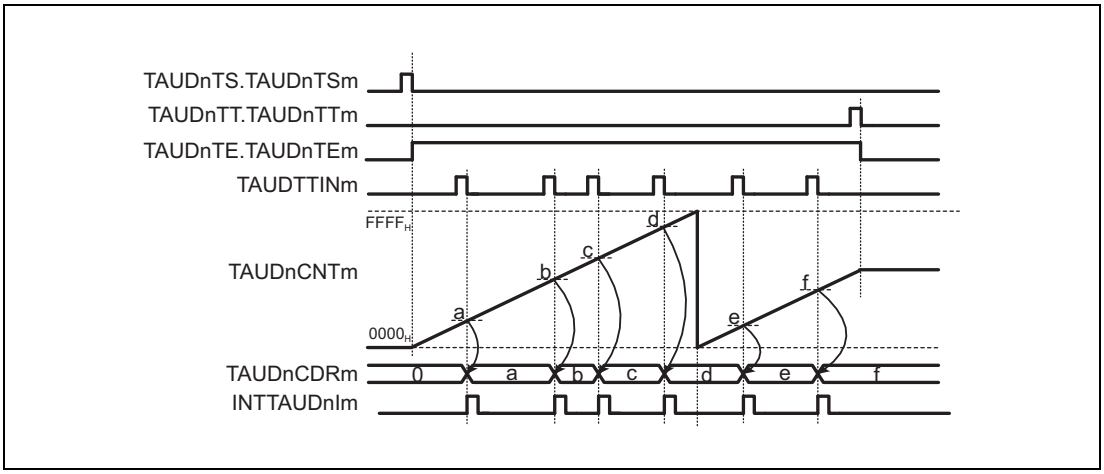



**Figure 25.65 General Timing Diagram of TAUDTTINm Input Position Detection Function**

### 25.12.9.4 Register Settings

#### (1) TAUDnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.87 Contents of the TAUDnCMORm Register for TAUDTTINm Input Position Detection Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation. Set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 001: Valid TAUDTTINm input edge signal is used as an external capture trigger. |
| 7, 6 | TAUDnCOS[1:0] | 01: Set to this value. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 1011: Count capture mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm not generated at the beginning of operation.<br>1: INTTAUDnIm generated at the beginning of operation. |

#### (2) TAUDnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.88 Contents of the TAUDnCMURm Register for TAUDTTINm Input Position Detection Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Detection of falling edge<br>01: Detection of rising edge<br>10: Detection of rising and falling edges<br>11: Setting prohibited |

#### (3) Channel output mode

The channel output mode is not used by this function.

### (4) Simultaneous rewrite

Simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the TAUDTTINm input position detection function. Therefore, these registers should be set to 0.

**Table 25.89 Simultaneous Rewrite Settings for TAUDTTINm Input Position Detection Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

## 25.12.9.5 Operating Procedure for TAUDTTINm Input Position Detection Function

**Table 25.90 Operating Procedure for TAUDTTINm Input Position Detection Function**

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers as described in **Table 25.87, Contents of the TAUDnCMORm Register for TAUDTTINm Input Position Detection Function**, and **Table 25.88, Contents of the TAUDnCMURm Register for TAUDTTINm Input Position Detection Function**.<br><br>The TAUDnCDRm register functions as a capture register. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is set to 1 and the counter starts.<br>If TAUDnCMORm.TAUDnMD0 is 1, INTTAUDnIm occurs. |
| During Operation | The values of TAUDnCMURm.TAUDnTIS[1:0] bits can be changed at any time.<br>The TAUDnCDRm and TAUDnCSRm registers can be read at any time. | TAUDnCNTm starts to count up from $0000_H$.<br>When a valid TAUDTTINm edge is detected:<br>• TAUDnCNTm transfers (captures) its own value to TAUDnCDRm.<br>• Outputs INTTAUDnIm.<br>• The counter is not cleared to $0000_H$ and TAUDnCNTm continues counting.<br>Afterwards, this procedure is repeated.<br>When TAUDnCNTm reaches $FFFF_H$, the counter restarts from $0000_H$. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops and retains its current value. |

Restart

### 25.12.9.6 Specific Timing Diagrams

#### (1) Operation stop and restart

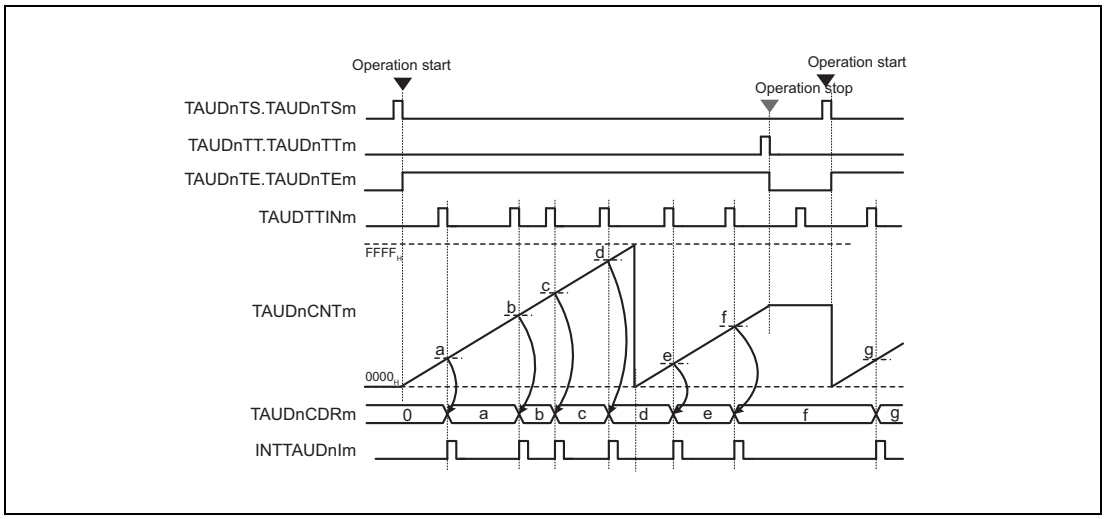



**Figure 25.66 Operation Stop and Restart (TAUDnCMORm.TAUDnMD0 = 0, TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)**

- The counter can stop operating by setting TAUDnTT.TAUDnTTm to 1. This sets TAUDnTE.TAUDnTEm to 0.
- TAUDnCNTm stops and retains its current value.
- If the counter stops operating, valid TAUDTTINm input edges are ignored.
- The counter can be restarted by setting TAUDnTS.TAUDnTSm to 1. TAUDnCNTm restarts to count from $0000_H$.

## 25.12.10 TAUDTTINm Input Period Count Detection Function

### 25.12.10.1 Overview

#### Summary

This function measures the cumulative width of a TAUDTTINm input signal.

#### Prerequisites

- The operating mode should be set to capture and gate count mode. (See **Table 25.91, Contents of the TAUDnCMORm Register for TAUDTTINm Input Period Count Detection Function**.)
- TAUDTTOUTm is not used with this function.

#### Functional description

The counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This in turn sets TAUDnTE.TAUDnTEm = 1, enabling count operation. The counter awaits a valid TAUDTTINm input edge.

When a valid TAUDTTINm input start edge is detected, the counter starts to count from $0000_H$.

When a valid TAUDTTINm input stop edge is detected, the current TAUDnCNTm value is loaded into TAUDnCDRm and an interrupt (INTTAUDnIm) is generated. The counter stops and retains its value (TAUDnCDRm + 1) until the next valid TAUDTTINm input start edge is detected.

When the next valid TAUDTTINm input start edge is detected, the counter restarts to count from the value retained when stopped.

If the counter reaches $FFFF_H$, the counter restarts from $0000_H$.

**NOTES**

1. TAUDTTINm input signal is sampled at the frequency of an operation clock set by the TAUDnCMORm.TAUDnCKS[1:0] bits.
2. As this function is to measure the TAUDTTINm input signal width, setting TAUDnTS.TAUDnTSm to 1 is disabled while TAUDnTE.TAUDnTEm = 1.

#### Conditions

The valid start and stop edges are specified by the TAUDnCMURm.TAUDnTIS[1:0] bits.

- If TAUDnCMURm.TAUDnTIS[1:0] = $10_B$, the TAUDTTINm input low width is measured. The start trigger is a falling edge and the stop trigger is a rising edge.
- If TAUDnCMURm.TAUDnTIS[1:0] = $11_B$, the TAUDTTINm input high width is measured. The start trigger is a rising edge and the stop trigger is a falling edge.

### 25.12.10.2 Equations

Cumulative TAUDTTINm input width =
count clock cycle × (TAUDnCDRm capture value + 1)

### 25.12.10.3 Block Diagram and General Timing Diagram

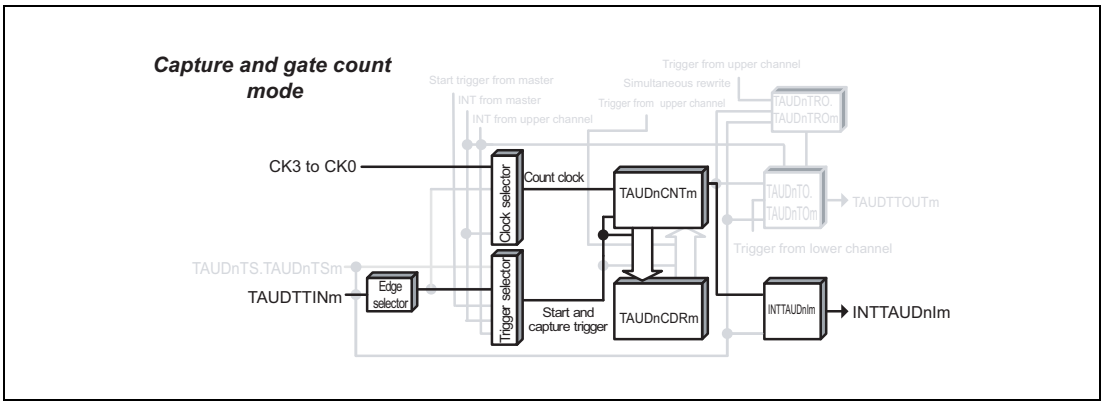



**Figure 25.67 Block Diagram of TAUDTTINm Input Period Count Detection Function**

The following settings apply to the general timing diagram.

- Detection of rising and falling edges = high width measurement (TAUDnCMURm.TAUDnTIS[1:0] = $11_B$)

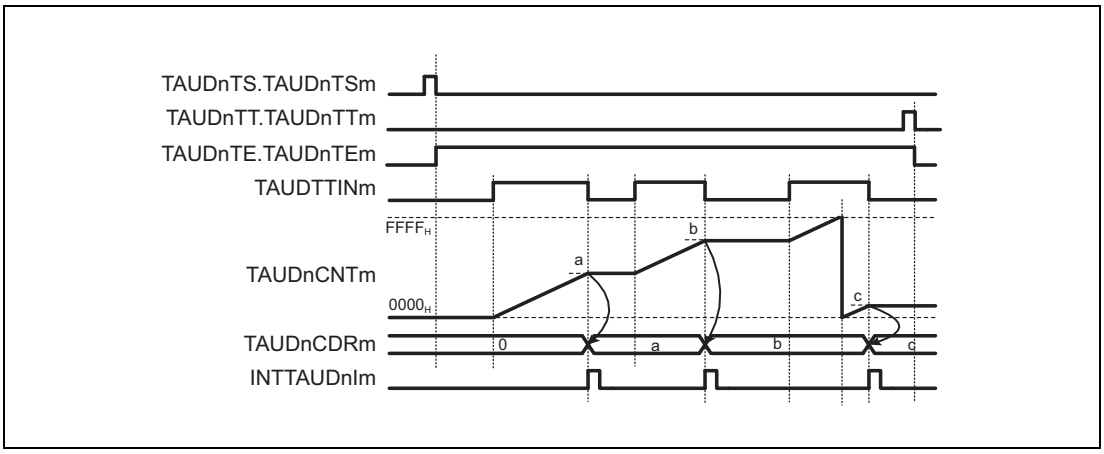



**Figure 25.68 General Timing Diagram of TAUDTTINm Input Period Count Detection Function**

### 25.12.10.4 Register Settings

#### (1) TAUDnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.91 Contents of the TAUDnCMORm Register for TAUDTTINm Input Period Count Detection Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation. Set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 010: Valid edge of the TAUDTTINm input signal is used as an external start trigger and the reverse edge as a stop trigger. |
| 7, 6 | TAUDnCOS[1:0] | 01: Set to this value. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 1101: Capture and gate count mode |
| 0 | TAUDnMD0 | 0: Disables the start trigger during operation. |

#### (2) TAUDnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.92 Contents of the TAUDnCMURm Register for TAUDTTINm Input Period Count Detection Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 10: Detection of rising and falling edges (low width measurement)<br>11: Detection of rising and falling edges (high width measurement) |

#### (3) Channel output mode

TAUDnTOE.TAUDnTOEm is set to 0 because the channel output mode is not used with this function.

#### (4) Simultaneous rewrite

Simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the TAUDTTINm input period count detection function. Therefore, these registers should be set to 0.

**Table 25.93 Simultaneous Rewrite Settings for TAUDTTINm Input Period Count Detection Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

### 25.12.10.5 Operating Procedure for TAUDTTINm Input Period Count Detection Function

**Table 25.94 Operating Procedure for TAUDTTINm Input Period Count Detection Function**

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers as described in **Table 25.91, Contents of the TAUDnCMORm Register for TAUDTTINm Input Period Count Detection Function**, and **Table 25.92, Contents of the TAUDnCMURm Register for TAUDTTINm Input Period Count Detection Function**.<br><br>The TAUDnCDRm register functions as a capture register. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is set to 1 and TAUDnCNTm waits for detection of the TAUDTTINm start edge. |
| During Operation | Detection of TAUDTTINm edge<br><br>The TAUDnCDRm, TAUDnCNTm, and TAUDnCSRm registers can be read at any time. | When a TAUDTTINm start edge (rising edge for high width measurement, falling edge for low width measurement) is detected, TAUDnCNTm starts counting up from the stop value.<br>When TAUDnCNTm detects a stop edge (falling edge for high width measurement, rising edge for low width measurement), it transfers the value to TAUDnCDRm and INTTAUDnIm is generated.<br>Counting stops at the “value transferred to TAUDnCDRm + 1” and TAUDnCNTm waits for detection of the TAUDTTINm start edge.<br>When TAUDnCNTm reaches $FFFF_H$, the counter restarts from $0000_H$.<br>Afterwards, this procedure is repeated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops and retains its current value. |

Restart Operation

### 25.12.10.6 Specific Timing Diagrams

#### (1) Operation stop and restart

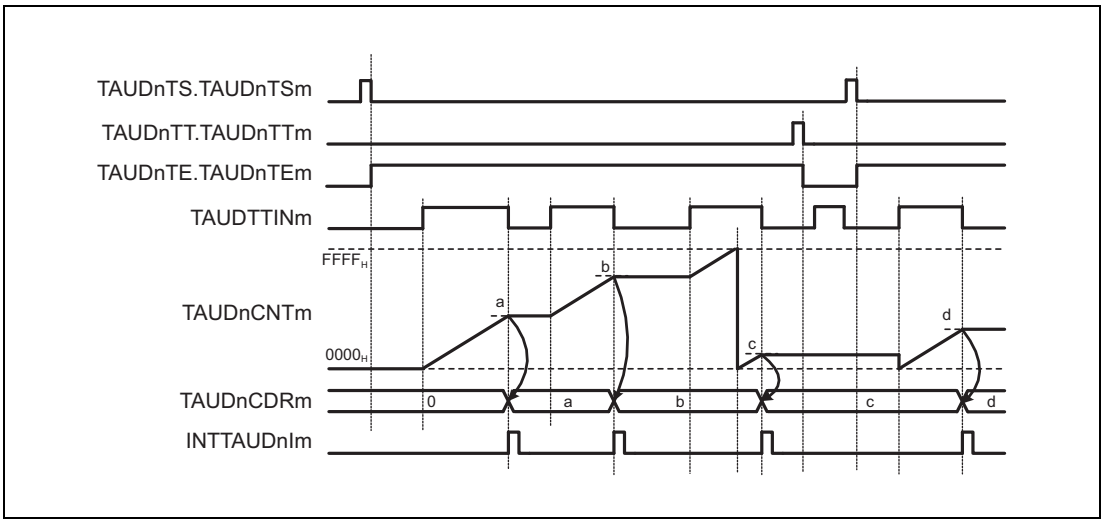



**Figure 25.69 Operation Stop and Restart (TAUDnCMURm.TAUDnTIS[1:0] = $11_B$)**

- The counter can be stopped by setting TAUDnTT.TAUDnTTm to 1. This sets TAUDnTE.TAUDnTEm to 0.
- TAUDnCNTm stops and retains its current value.
- If the counter is stopped, valid TAUDTTINm input edges are ignored.
- The counter can be restarted by setting TAUDnTS.TAUDnTSm to 1. TAUDnCNTm restarts to count from $0000_H$.

## 25.12.11 TAUDTTINm Input Pulse Interval Judgment Function

### 25.12.11.1 Overview

#### Summary

This function outputs the result of a comparison between the count value (TAUDnCNTm) and the value in the channel data register (TAUDnCDRm) when a TAUDTTINm input pulse occurs. An interrupt request signal INTTAUDnIm is generated if the result of the comparison is true.

#### Prerequisites

- The operating mode should be set to judge mode. See **Table 25.95, Contents of the TAUDnCMORm Register for TAUDTTINm Input Pulse Interval Judgment Function**.
- TAUDTTOUTm is not used with this function.

#### Functional description

The counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This in turn sets TAUDnTE.TAUDnTEm = 1, enabling count operation. The current value of TAUDnCDRm is loaded into TAUDnCNTm and the counter starts to count down from this value.

When a TAUDTTINm valid edge is detected or TAUDnTS.TAUDnTSm is set to 1, the function compares the current values of TAUDnCNTm and TAUDnCDRm. An interrupt request signal INTTAUDnIm is generated if the result of the comparison is true. TAUDnCNTm reloads the value of TAUDnCDRm and subsequently continues operation, regardless of the result of the comparison.

If the counter reaches $0000_H$ before a TAUDTTINm valid edge is detected, TAUDnCNTm overflows and is set to $FFFF_H$. It then continues to count down.

The value of TAUDnCDRm can be rewritten at any time, and the changed value of TAUDnCDRm is applied the next time the counter starts to count down.

#### Conditions

The TAUDnCMORm.TAUDnMD0 bit specifies the type of comparison:

- If TAUDnCMORm.TAUDnMD0 = 0, INTTAUDnIm is generated when TAUDnCNTm ≤ TAUDnCDRm.
- If TAUDnCMORm.TAUDnMD0 = 1, INTTAUDnIm is generated when TAUDnCNTm > TAUDnCDRm.

### 25.12.11.2 Block Diagram and General Timing Diagram

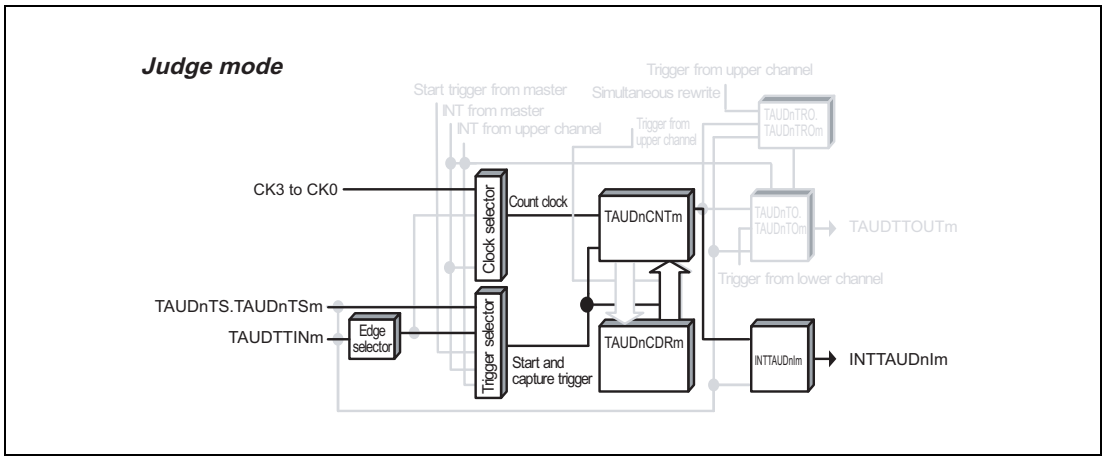



**Figure 25.70 Block Diagram of TAUDTTINm Input Pulse Interval Judgment Function**

The following settings apply to the general timing diagram.

- Detection of falling edge (TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)

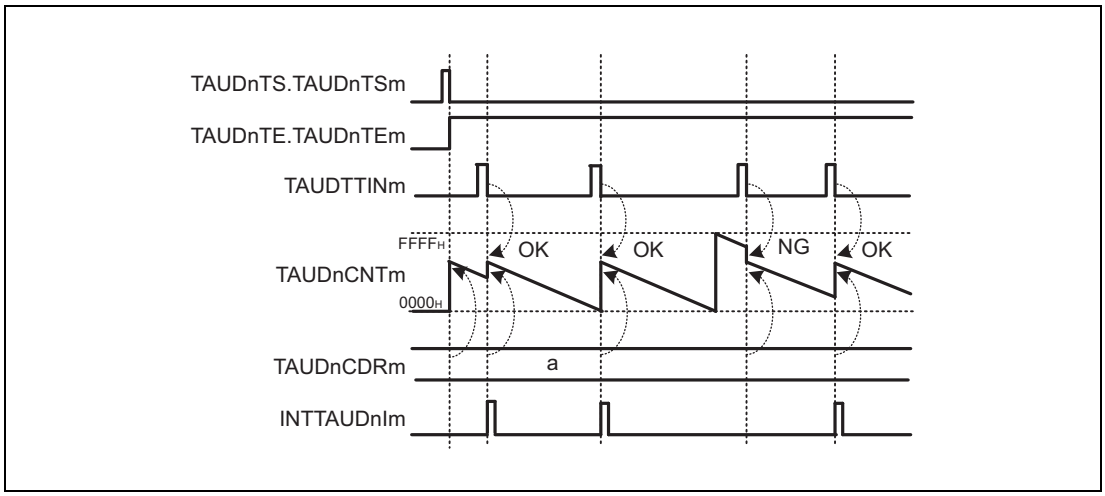



**Figure 25.71 General Timing Diagram of TAUDTTINm Input Pulse Interval Judgment Function**

### 25.12.11.3 Register Settings

#### (1) TAUDnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.95 Contents of the TAUDnCMORm Register for TAUDTTINm Input Pulse Interval Judgment Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation, set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 001: Valid edge of the TAUDTTINm input signal is used as an external start trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0001: Judge mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm is generated when TAUDnCNTm ≤ TAUDnCDRm<br>1: INTTAUDnIm is generated when TAUDnCNTm > TAUDnCDRm |

#### (2) TAUDnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.96 Contents of the TAUDnCMURm Register for TAUDTTINm Input Pulse Interval Judgment Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Detection of falling edge<br>01: Detection of rising edge<br>10: Detection of rising and falling edges<br>11: Setting prohibited |

### (3) Channel output mode

TAUDnTOE.TAUDnTOEm is set to 0 because the channel output mode is not used with this function.

### (4) Simultaneous rewrite

Simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the TAUDTTINm input pulse interval judgment function. Therefore, these registers should be set to 0.

**Table 25.97 Simultaneous Rewrite Settings for TAUDTTINm Input Pulse Interval Judgment Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

## 25.12.11.4 Operating Procedure for TAUDTTINm Input Pulse Interval Judgment Function

**Table 25.98 Operating Procedure for TAUDTTINm Input Pulse Interval Judgment Function**

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers as described in **Table 25.95, Contents of the TAUDnCMORm Register for TAUDTTINm Input Pulse Interval Judgment Function**, and **Table 25.96, Contents of the TAUDnCMURm Register for TAUDTTINm Input Pulse Interval Judgment Function**.<br><br>Set the value of TAUDnCDRm register. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is set to 1 and the counter starts.<br>TAUDnCDRm value is loaded into TAUDnCNTm. |
| During Operation | The following register can be changed at any time:<br>• TAUDnCDRm register | When TAUDnCMORm.TAUDnMD0 = 0<br>If TAUDnCNTm ≤ TAUDnCDRm when a TAUDTTINm input edge is detected, INTTAUDnIm is generated.<br>When TAUDnCMORm.TAUDnMD0 = 1<br>If TAUDnCNTm > TAUDnCDRm when a TAUDTTINm input edge is detected, INTTAUDnIm is generated. If a TAUDTTINm input edge is detected, then TAUDnCNTm starts to count down from the value of TAUDnCDRm. Afterwards, this procedure is repeated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops and retains its current value. |

Restart Operation

## 25.12.12 TAUDTTINm Input Signal Width Judgment Function

### 25.12.12.1 Overview

#### Summary

This function compares the count value (TAUDnCNTm) for the high or low level width of a TAUDTTINm input signal and the TAUDnCDRm value, and outputs the judgment result from the interrupt request signal INTTAUDnIm.

#### Prerequisites

- The operating mode should be set to judge and one-count mode. (See **Table 25.99, Contents of the TAUDnCMORm Register for TAUDTTINm Input Signal Width Judgment Function**.)
- TAUDTTOUTm is not used with this function.

#### Functional description

The counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This in turn sets TAUDnTE.TAUDnTEm = 1, enabling count operation. When a valid TAUDTTINm input start edge is detected, the current value of TAUDnCDRm is loaded into TAUDnCNTm and the counter starts to count down from this value.

When a TAUDTTINm valid stop edge is detected, the function compares the current values of TAUDnCNTm and TAUDnCDRm. An interrupt request signal INTTAUDnIm is generated if the result of the comparison is true. The counter TAUDnCNTm retains its value until the next valid TAUDTTINm start edge is detected, regardless of the result of the comparison.

If the counter reaches $0000_H$ before a valid TAUDTTINm stop edge is detected, TAUDnCNTm overflows and is set to $FFFF_H$. The counter then continues to count down.

The value of TAUDnCDRm can be rewritten at any time, and the changed value of TAUDnCDRm is applied the next time the counter starts to count down.

#### Conditions

- The TAUDnCMORm.TAUDnMD0 bit specifies the type of comparison:
  - If TAUDnCMORm.TAUDnMD0 = 0, INTTAUDnIm is generated when TAUDnCNTm ≤ TAUDnCDRm.
  - If TAUDnCMORm.TAUDnMD0 = 1, INTTAUDnIm is generated when TAUDnCNTm > TAUDnCDRm.
- The TAUDnCMURm.TAUDnTIS[1:0] bits specify a type of width measurement:
  - For high width measurement (TAUDnCMURm.TAUDnTIS[1:0] = $11_B$), TAUDTTINm rising edge is used as a start edge and TAUDTTINm falling edge as a stop edge.
  - For low width measurement (TAUDnCMURm.TAUDnTIS[1:0] = $10_B$), TAUDTTINm falling edge is used as a start edge and TAUDTTINm rising edge as a stop edge.
- This function cannot make a forced restart.

### 25.12.12.2 Block Diagram and General Timing Diagram

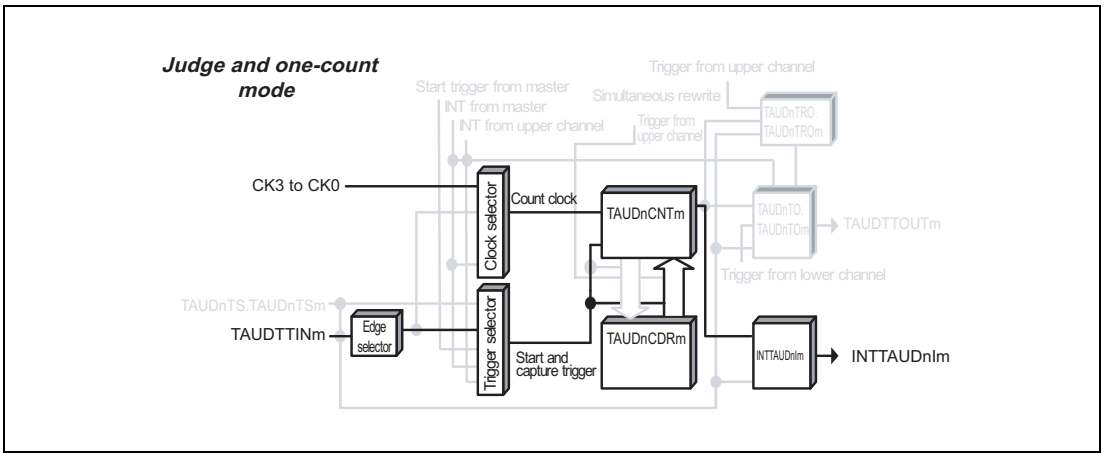



**Figure 25.72 Block Diagram of TAUDTTINm Input Signal Width Judgment Function**

The following settings apply to the general timing diagram.

- INTTAUDnIm is generated when TAUDnCNTm ≤ TAUDnCDRm (TAUDnCMORm.TAUDnMD0 = 0).
- TAUDTTINm valid start edge = rising edge, TAUDTTINm valid stop edge = falling edge (TAUDnCMURm.TAUDnTIS[1:0] = $11_B$)

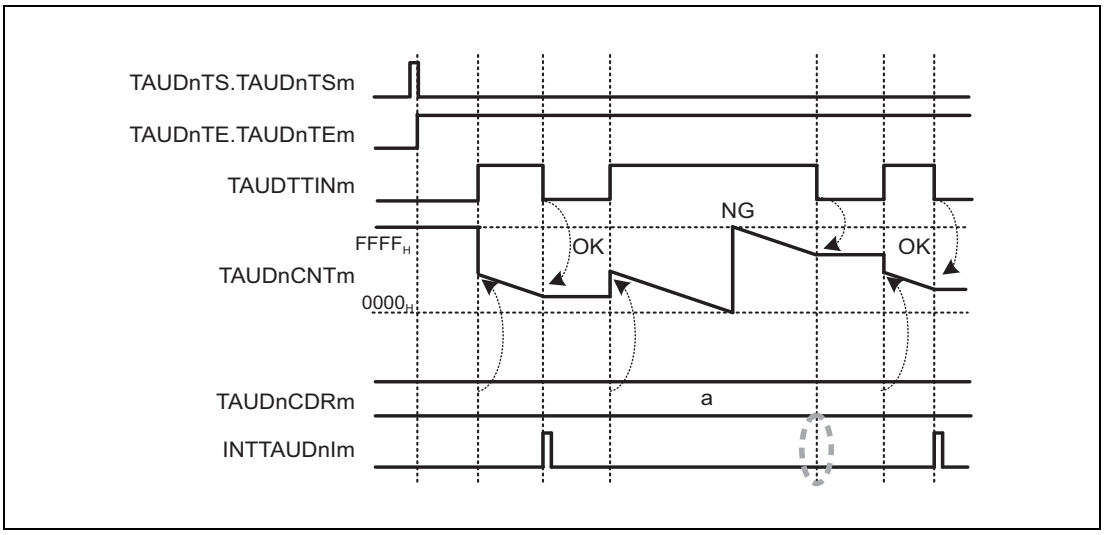



**Figure 25.73 General Timing Diagram of TAUDTTINm Input Signal Width Judgment Function**

### 25.12.12.3 Register Settings

#### (1) TAUDnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.99 Contents of the TAUDnCMORm Register for TAUDTTINm Input Signal Width Judgment Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation, set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 010: Valid edge of the TAUDTTINm input signal is used as an external start trigger and the reverse edge as a stop trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0111: Judge and one-count mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm is generated when TAUDnCNTm ≤ TAUDnCDRm<br>1: INTTAUDnIm is generated when TAUDnCNTm > TAUDnCDRm |

#### (2) TAUDnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.100 Contents of the TAUDnCMURm Register for TAUDTTINm Input Signal Width Judgment Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 10: Detection of rising and falling edges (low width measurement)<br>11: Detection of rising and falling edges (high width measurement) |

#### (3) Channel output mode

TAUDnTOE.TAUDnTOEm is set to 0 because the channel output mode is not used with this function.

#### (4) Simultaneous rewrite

Simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the TAUDTTINm input signal width judgment function. Therefore, these registers should be set to 0.

**Table 25.101 Simultaneous Rewrite Settings for TAUDTTINm Input Signal Width Judgment Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

### 25.12.12.4 Operating Procedure for TAUDTTINm Input Signal Width Judgment Function

**Table 25.102 Operating Procedure for TAUDTTINm Input Signal Width Judgment Function**

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers as described in **Table 25.99, Contents of the TAUDnCMORm Register for TAUDTTINm Input Signal Width Judgment Function**, and **Table 25.100, Contents of the TAUDnCMURm Register for TAUDTTINm Input Signal Width Judgment Function**.<br><br>Set the value of TAUDnCDRm register. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is set to 1 and TAUDnCNTm waits for detection of the TAUDTTINm start edge. |
| During Operation | The following register can be changed at any time:<br>• TAUDnCDRm register | Upon detection of a TAUDTTINm start edge, TAUDnCNTm starts count down from the value of TAUDnCDRm.<br><br>When TAUDnCMORm.TAUDnMD0 = 0<br>If TAUDnCNTm ≤ TAUDnCDRm when a TAUDTTINm input stop edge is detected, INTTAUDnIm is generated.<br><br>When TAUDnCMORm.TAUDnMD0 = 1<br>If TAUDnCNTm > TAUDnCDRm when a TAUDTTINm input stop edge is detected, INTTAUDnIm is generated.<br><br>Afterwards, this procedure is repeated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops and retains its current value. |

Restart Operation

## 25.12.13 Overflow Interrupt Output Function (during TAUDTTINm Width Measurement)

### 25.12.13.1 Overview

#### Summary

This function measures the width of an individual TAUDTTINm input signal. An interrupt is generated if the TAUDTTINm input width is longer than $FFFF_H + 1$.

#### Prerequisites

- The operation mode must be set to One-Count Mode (see **Table 25.103, Contents of the TAUDnCMORm Register for Overflow Interrupt Output Function (during TAUDTTINm Width Measurement)**).
- TAUDTTOUTm is not used for this function.
- The value of TAUDnCDRm must be set to $FFFF_H$.

#### Functional description

The counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This in turn sets TAUDnTE.TAUDnTEm = 1, enabling count operation.

The counter starts when a valid TAUDTTINm input start edge is detected. $FFFF_H$ is loaded to TAUDnCNTm and the counter starts to count down.

When a valid stop edge is detected, the counter stops and retains the current value.

When the next TAUDTTINm input start edge is detected, TAUDnCNTm loads $FFFF_H$ and starts to count down.

If the counter reaches $0000_H$ before a stop edge is detected, an interrupt is generated.

#### Conditions

The valid start and stop edges are specified by the TAUDnCMURm.TAUDnTIS[1:0] bits.

- If TAUDnCMURm.TAUDnTIS[1:0] = $10_B$, the TAUDTTINm input low width is measured. The start trigger is a falling edge and the stop trigger is a rising edge.
- If TAUDnCMURm.TAUDnTIS[1:0] = $11_B$, the TAUDTTINm input high width is measured. The start trigger is a rising edge and the stop trigger is a falling edge.

**NOTE**

The counter cannot be restarted during operation.

### 25.12.13.2 Block Diagram and General Timing Diagram

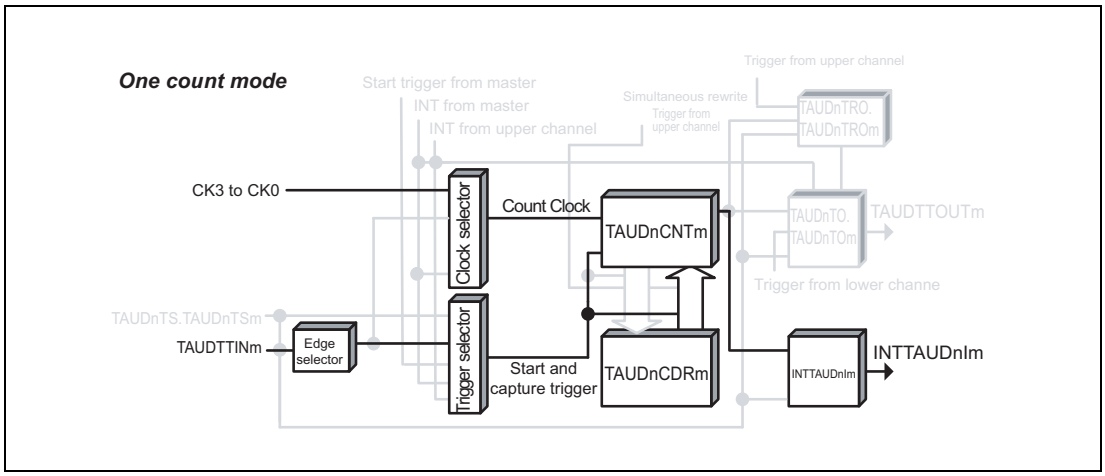



**Figure 25.74 Block Diagram for Overflow Interrupt Output Function (during TAUDTTINm Width Measurement)**

The following settings apply to the general timing diagram.

- Detection of rising and falling edges = high width measurement (TAUDnCMURm.TAUDnTIS[1:0] = $11_B$)

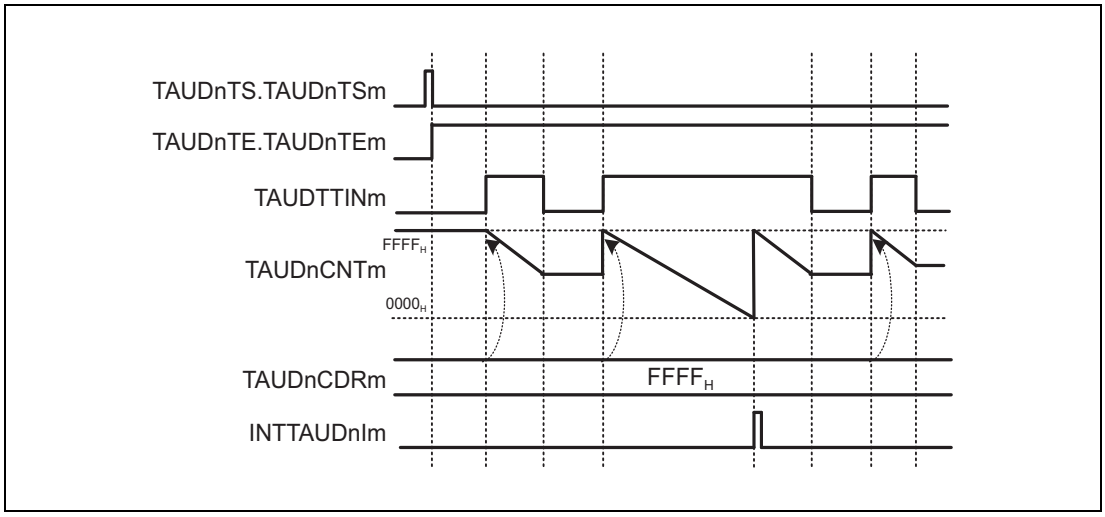



**Figure 25.75 General Timing Diagram for Overflow Interrupt Output Function (during TAUDTTINm Width Measurement)**

### 25.12.13.3 Register Settings

#### (1) TAUDnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.103 Contents of the TAUDnCMORm Register for Overflow Interrupt Output Function (during TAUDTTINm Width Measurement)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation. Set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 010: Valid edge of the TAUDTTINm input signal is used as an external start trigger and the reverse edge as a stop trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0100: One-count mode |
| 0 | TAUDnMD0 | 0: Disables the start trigger during operation |

#### (2) TAUDnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.104 Contents of the TAUDnCMURm Register for Overflow Interrupt Output Function (during TAUDTTINm Width Measurement)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 10: Detection of rising and falling edges (low width measurement)<br>11: Detection of rising and falling edges (high width measurement) |

#### (3) Channel output mode

TAUDnTOE.TAUDnTOEm is set to 0 because the channel output mode is not used by this function.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the Overflow Interrupt Output Function (during TAUDTTINm Width Measurement). Therefore, these registers must be set to 0.

**Table 25.105 Simultaneous Rewrite Settings for Overflow Interrupt Output Function (during TAUDTTINm Width Measurement)**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

## 25.12.13.4 Operating Procedure for Overflow Interrupt Output Function (during TAUDTTINm Width Measurement)

**Table 25.106 Operating Procedure for Overflow Interrupt Output Function (during TAUDTTINm Width Measurement)**

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers as described in **Table 25.103, Contents of the TAUDnCMORm Register for Overflow Interrupt Output Function (during TAUDTTINm Width Measurement)**, and **Table 25.104, Contents of the TAUDnCMURm Register for Overflow Interrupt Output Function (during TAUDTTINm Width Measurement)**.<br><br>Set the value of TAUDnCDRm register to $FFFF_H$. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0.<br><br>Detection of TAUDTTINm start edge | TAUDnTE.TAUDnTEm is set to 1 and TAUDnCNTm waits for detection of the start edge.<br><br>When a start edge is detected, TAUDnCNTm loads the TAUDnCDRm value ($FFFF_H$). |
| During Operation | The TAUDnCNTm register can be read at any time. | TAUDnCNTm counts down. When the counter reaches $0000_H$:<br>• INTTAUDnIm is generated.<br>When TAUDTTINm input stop edge is detected during count operation:<br>• TAUDnCNTm stops and retains its current value.<br>When TAUDTTINm input start edge is detected while the counter is stopped:<br>• TAUDnCNTm loads the TAUDnCDRm value ($FFFF_H$) again, and continues to count down.<br>Afterwards, this procedure is repeated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops and retains its current value. |

Restart Operation

## 25.12.14 Overflow Interrupt Output Function (during TAUDTTINm Input Period Count Detection)

### 25.12.14.1 Overview

#### Summary

This function measures the cumulative width of a TAUDTTINm input signal. If the cumulative TAUDTTINm input width is longer than $FFFF_H$, an interrupt is generated and an overflow interrupt can be output.

#### Prerequisites

- The operation mode must be set to Gate Count Mode, (see **Table 25.107, Contents of the TAUDnCMORm Register for Overflow Interrupt Output Function (during TAUDTTINm Input Period Count Detection)**).
- TAUDTTOUTm is not used with this function.
- The value of TAUDnCDRm must be set to $FFFF_H$.

#### Functional description

The counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This in turn sets TAUDnTE.TAUDnTEm = 1, enabling count operation.

The counter starts when a valid TAUDTTINm input start edge is detected. $FFFF_H$ is loaded to TAUDnCNTm and the counter starts to count down.

When a valid stop edge is detected, the counter stops and retains the current value. The counter awaits the next TAUDTTINm input start edge and then continues to count down from the current value.

When the counter reaches $0000_H$ an interrupt is generated. $FFFF_H$ is loaded to TAUDnCNTm and the counter continues to count down until a TAUDTTINm input stop edge is detected.

#### Conditions

The valid start and stop edges are specified by the TAUDnCMURm.TIS[1:0] bits.

- If TAUDnCMURm.TAUDnTIS[1:0] = $10_B$, the TAUDTTINm input low width is counted. The start trigger is a falling edge and the stop trigger is a rising edge.
- If TAUDnCMURm.TAUDnTIS[1:0] = $11_B$, the TAUDTTINm input high width is counted. The start trigger is a rising edge and the stop trigger is a falling edge.

**NOTE**

The counter cannot be restarted during operation.

### 25.12.14.2 Block Diagram and General Timing Diagram

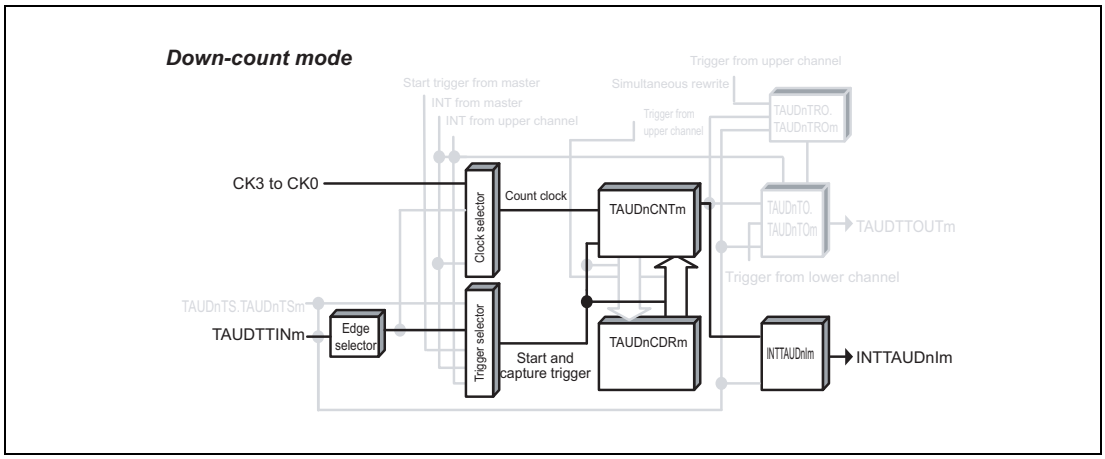



**Figure 25.76 Block Diagram for Overflow Interrupt Output Function (during TAUDTTINm Input Period Count Detection)**

The following settings apply to the general timing diagram.

- Detection of rising and falling edges = high width measurement (TAUDnCMURm.TAUDnTIS[1:0] = $11_B$)

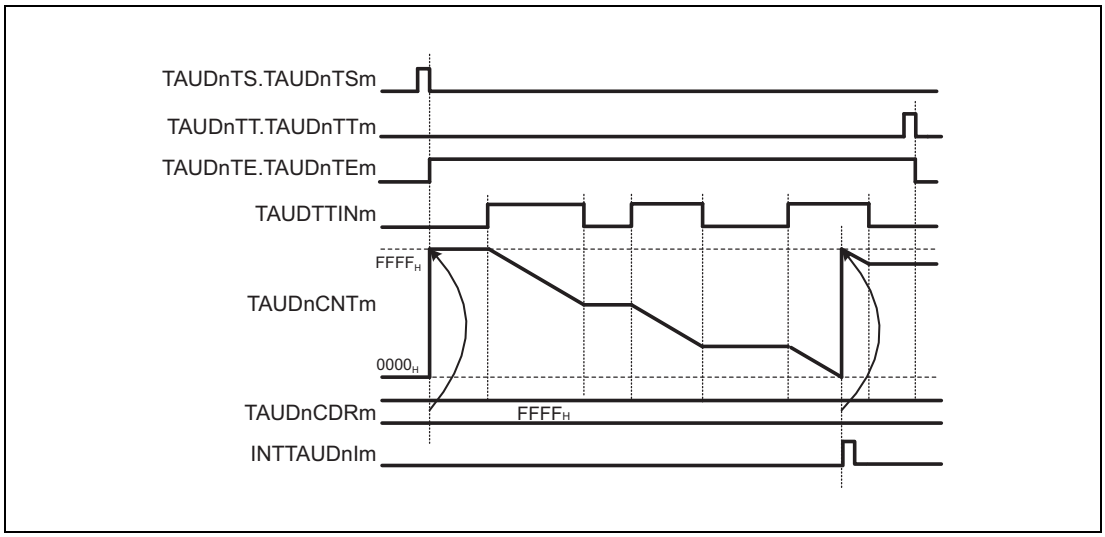



**Figure 25.77 General Timing Diagram for Overflow Interrupt Output Function (during TAUDTTINm Input Period Count Detection)**

### 25.12.14.3 Register Settings

#### (1) TAUDnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.107 Contents of the TAUDnCMORm Register for Overflow Interrupt Output Function (during TAUDTTINm Input Period Count Detection)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation. Set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 010: Valid edge of the TAUDTTINm input signal is used as an external start trigger and the reverse edge as a stop trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 1100: Gate count mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm not generated at the beginning of operation |

#### (2) TAUDnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.108 Contents of the TAUDnCMURm Register for Overflow Interrupt Output Function (during TAUDTTINm Input Period Count Detection)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing , write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 10: Detection of rising and falling edges (low width measurement)<br>11: Detection of rising and falling edges (high width measurement) |

#### (3) Channel output mode

TAUDnTOE.TAUDnTOEm is set to 0 because the channel output mode is not used with this function.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the Overflow Interrupt Output Function (during TAUDTTINm Input Period Count Detection). Therefore, these registers must be set to 0.

**Table 25.109 Simultaneous Rewrite Settings for Overflow Interrupt Output Function (during TAUDTTINm Input Period Count Detection)**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

## 25.12.14.4 Operating Procedure for Overflow Interrupt Output Function (during TAUDTTINm Input Period Count Detection)

**Table 25.110 Operating Procedure for Overflow Interrupt Output Function (during TAUDTTINm Input Period Count Detection)**

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers as described in **Table 25.107, Contents of the TAUDnCMORm Register for Overflow Interrupt Output Function (during TAUDTTINm Input Period Count Detection)**, and **Table 25.108, Contents of the TAUDnCMURm Register for Overflow Interrupt Output Function (during TAUDTTINm Input Period Count Detection)**.<br><br>Set the value of TAUDnCDRm register to $FFFF_H$. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0.<br><br>Detection of TAUDTTINm start edge | TAUDnTE.TAUDnTEm is set to 1 and TAUDnCNTm waits for detection of the start edge.<br><br>When a start edge is detected, TAUDnCNTm loads the TAUDnCDRm value ($FFFF_H$). |
| During Operation | The TAUDnCNTm register can be read at all times. | TAUDnCNTm counts down. When the counter reaches $0000_H$:<br>• INTTAUDnIm is generated.<br>• TAUDnCNTm loads the TAUDnCDRm value ($FFFF_H$) and continues to count down.<br>When TAUDTTINm input stop edge is detected during count operation:<br>• TAUDnCNTm stops and retains the current value.<br>When TAUDTTINm input start edge is detected while the counter is stopped:<br>• TAUDnCNTm counts down from the stop value.<br>Afterwards, this procedure is repeated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops and retains its current value. |

Restart Operation

## 25.12.15 One-Phase PWM Output Function

### 25.12.15.1 Overview

#### Summary

This function adds dead time to a TAUDTTINm input signal. The resulting PWM signal is output via TAUDTTOUTm of the channel and TAUDTTOUTm of upper channels.

#### Prerequisites

- Each of two (or more) channels is enabled for dead time control (TAUDnTDE.TAUDnTDEm = 1).
- The operating mode for the lower channel should be set to one-count mode. (See **Table 25.112, Contents of the TAUDnCMORm Register for the Lower Channel of the One-Phase PWM Output Function**.)
- Any operating mode can be set to upper channels.
- Channel output mode for upper and lower channels should be set to synchronous channel output mode 2 with one-phase PWM output. (See **Section 25.7, Channel Output Modes**.)

#### Functional description

The counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This sets TAUDnTE.TAUDnTEm = 1, enabling count operation.

The counter starts when a valid TAUDTTINm input start edge is detected. The value of TAUDnCDRm is loaded into TAUDnCNTm and the counter starts to count down from the TAUDnCDRm value.

When the counter reaches $0000_H$, an interrupt occurs. The counter is reset to $FFFF_H$ and waits for the next valid TAUDTTINm input start edge.

**Table 25.111 TAUDTTOUTm to which Dead Time is Added and State of TAUDTTINm**

| TAUDnCMURm. TAUDnTIS[1:0] | TAUDnTOL. TAUDnTOLm | TAUDTTOUTm to which Dead Time is Added | TAUDnTDL. TAUDnTDLm | TAUDTTINm State when Added |
|---|---|---|---|---|
| 10 | 0 | TAUDTTOUTm low | 0 | High |
| | | | 1 | Low |
| | 1 | TAUDTTOUTm high | 0 | High |
| | | | 1 | Low |
| 11 | 0 | TAUDTTOUTm low | 0 | Low |
| | | | 1 | High |
| | 1 | TAUDTTOUTm high | 0 | Low |
| | | | 1 | High |

**Conditions**

- TAUDnCMURm.TAUDnTIS[1:0] bits specify the type of width measurement:
  - TAUDnCMURm.TAUDnTIS[1:0] = $10_B$: Uses both edges as valid edges for detection (Low width measurement).
  - TAUDnCMURm.TAUDnTIS[1:0] = $11_B$: Uses both edges as valid edges for detection (High width measurement).
- The TAUDnTDL.TAUDnTDLm bit specifies the operation of TAUDTTOUTm for each channel when an interrupt or valid TAUDTTINm edge is detected on the lower channel:
  - If TAUDnTDL.TAUDnTDLm = 0, an interrupt is used as a TAUDTTOUTm set trigger and a valid TAUDTTINm edge as a TAUDTTOUTm reset trigger.
  - If TAUDnTDL.TAUDnTDLm = 1, a valid TAUDTTINm edge is used as a TAUDTTOUTm set trigger and an interrupt as a TAUDTTOUTm reset trigger.
- This function cannot make a forced restart.

### 25.12.15.2 Block Diagram and General Timing Diagram

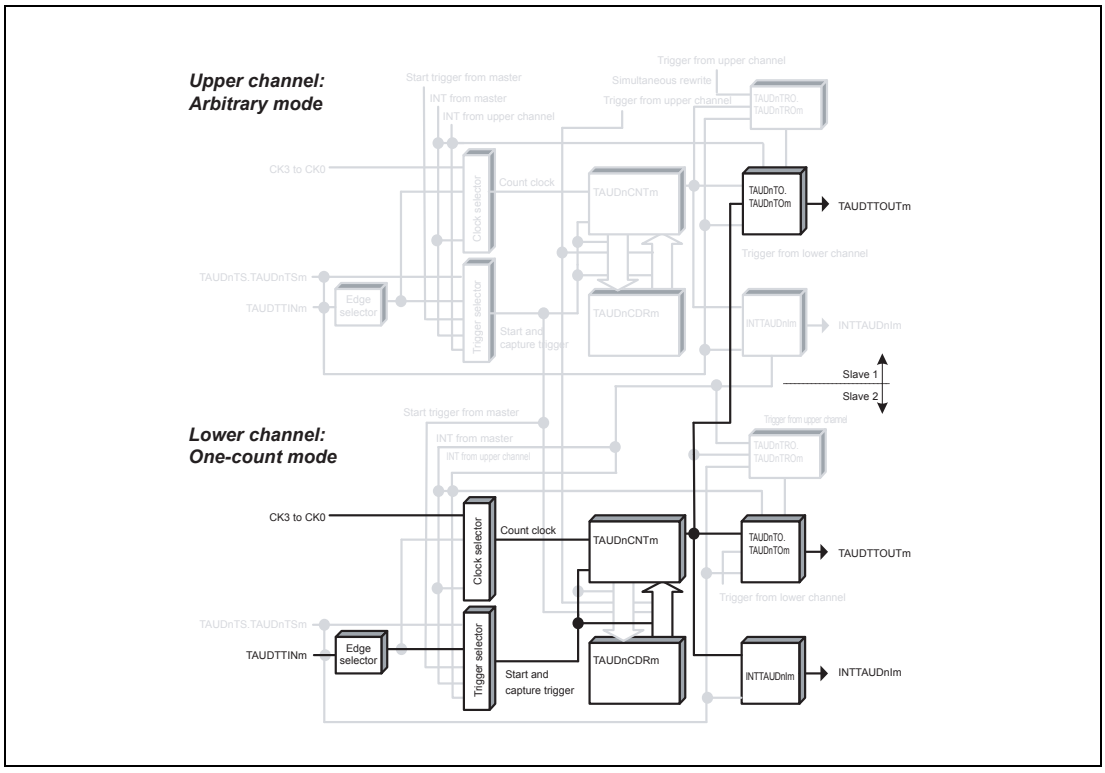



**Figure 25.78 Block Diagram of One-Phase PWM Output Function**

The following settings apply to the general timing diagram.

- Detection of rising and falling edges = high width measurement (TAUDnCMURm.TAUDnTIS[1:0] = $11_B$)

This setting considers a duty as an active high.

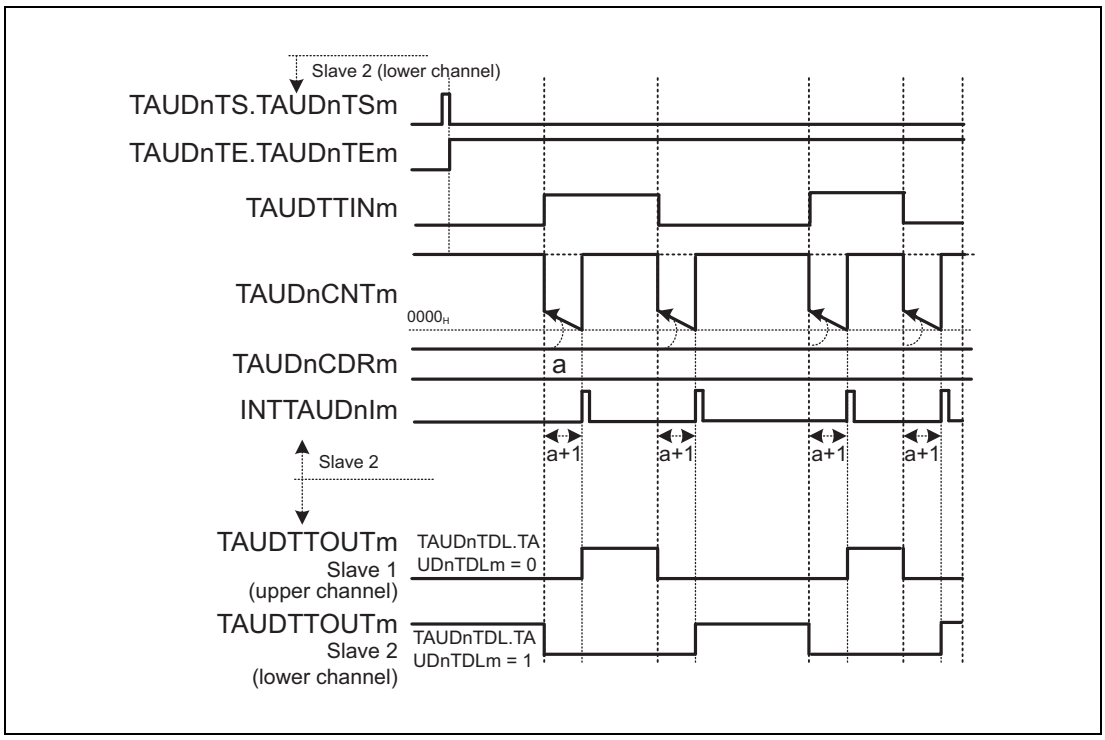



**Figure 25.79 General Timing Diagram of One-Phase PWM Output Function**

### 25.12.15.3 Register Settings for Lower Channels

#### (1) TAUDnCMORm for lower channels

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.112 Contents of the TAUDnCMORm Register for the Lower Channel of the One-Phase PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation, set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 001: Valid edge of the TAUDTTINm input signal is used as an external start trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0100: One-count mode |
| 0 | TAUDnMD0 | 1: Enables start trigger detection while counting. |

#### (2) TAUDnCMURm for lower channels

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.113 Contents of the TAUDnCMURm Register for the Lower Channel of the One-Phase PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 10: Detection of rising and falling edges (low width measurement)<br>11: Detection of rising and falling edges (high width measurement) |

### (3) Channel output mode for lower channels

Table 25.114 Control Bit Settings in Synchronous Channel Output Mode 2 with One-Phase PWM Output

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 1: Synchronous channel output |
| TAUDnTOC.TAUDnTOCm | 1: Operating mode 2 |
| TAUDnTOL.TAUDnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUDnTDE.TAUDnTDEm | 1: Enables dead time operation. |
| TAUDnTDM.TAUDnTDMm | 1: Adds dead time upon detection of a TAUDTTINm input edge on a lower odd channel. |
| TAUDnTDL.TAUDnTDLm | 0: Adds dead time of the positive-phase width<br>1: Adds dead time of the negative-phase width |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set this bit to 0 |
| TAUDnTRC.TAUDnTRCm | 0: Disables the operation as a real-time output trigger channel |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

**CAUTION**

Set TAUDnTDL.TAUDnTDLm exclusively from upper channels.

### (4) Simultaneous rewrite for lower channels

Simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the one-phase PWM output function. Therefore, these registers should be set to 0.

Table 25.115 Simultaneous Rewrite Settings for One-Phase PWM Output Function

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

### 25.12.15.4 Register Settings for Upper Channels

#### (1) TAUDnCMORm for upper channels

TAUDnCMORm register for upper channels can be set arbitrarily.

#### (2) TAUDnCMURm for upper channels

TAUDnCMURm register for upper channels can be set arbitrarily.

#### (3) Channel output mode for upper channels

**Table 25.116 Control Bit Settings for Upper Channels in Synchronous Channel Output Mode 2 with One-Phase PWM Output**

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 1: Synchronous channel output |
| TAUDnTOC.TAUDnTOCm | 1: Operating mode 2 |
| TAUDnTOL.TAUDnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUDnTDE.TAUDnTDEm | 1: Enables dead time operation. |
| TAUDnTDM.TAUDnTDMm | 1: Adds dead time upon detection of a TAUDTTINm input edge on a lower odd channel. |
| TAUDnTDL.TAUDnTDLm | 0: Adds dead time of the positive-phase width<br>1: Adds dead time of the negative-phase width |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set this bit to 0 |
| TAUDnTRC.TAUDnTRCm | 0: Disables the operation as a real-time output trigger channel |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

**CAUTION**

Set TAUDnTDL.TAUDnTDLm exclusively from lower channels.

#### (4) Simultaneous rewrite for upper channels

Simultaneous rewrite register for upper channels can be set arbitrarily.

### 25.12.15.5 Operating Procedure for One-phase PWM Output Function

Table 25.117 Operating Procedure for One-phase PWM Output Function

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers for the lower channel as described in **Table 25.112, Contents of the TAUDnCMORm Register for the Lower Channel of the One-Phase PWM Output Function**, and **Table 25.113, Contents of the TAUDnCMURm Register for the Lower Channel of the One-Phase PWM Output Function**.<br><br>Set TAUDnCMORm and TAUDnCMURm registers for the upper channel as described in **Section 25.12.15.4, Register Settings for Upper Channels**.<br><br>Set the value of TAUDnCDRm register.<br><br>Set channel output mode by setting the control bits as described in **Table 25.114, Control Bit Settings in Synchronous Channel Output Mode 2 with One-Phase PWM Output**. | Channel operation is stopped. |
| Start Operation | Set TAUDnTOE.TAUDnTOEm (slave channels 1 and 2) to 1 (at restart time only).<br>Set TAUDnTS.TAUDnTSm = 1 for slave channel 2.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0.<br>Detection of TAUDTTINm start edge | TAUDnTE.TAUDnTEm is set to 1 (slave channel 2) and TAUDnCNTm waits for detection of TAUDTTINm start edge.<br><br>TAUDnCNTm loads TAUDnCDRm value. |
| During Operation | The TAUDnCDRm register value can be changed at any time.<br>The TAUDnCNTm register can be read at any time. | TAUDnCNTm of slave channel 2 counts down. When the counter reaches $0000_H$:<br>• INTTAUDnIm is generated.<br>• TAUDnCNTm stops counting.<br><br>TAUDTTOUTm is changed by a TAUDTTINm edge detection signal and slave channel 2 INTTAUDnIm signal to output one-phase PWM waveform with dead time.<br>Afterwards, this operation is repeated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm = 1 for slave channel 2.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops. TAUDnCNTm and TAUDTTOUTm retain their current values. |

Restart Operation

# 25.13 Independent Channel Real-Time Functions

This section describes functions that output the value of the TAUDnTRO.TAUDnTROm bit in real time.

## 25.13.1 Real-Time Output Function Type 1

### 25.13.1.1 Overview

#### Summary

This function outputs a value of the TAUDnTRO.TAUDnTROm bit from TAUDTTOUTm when a specified channel generates an interrupt (INTTAUDnIm). In this function, the interrupt is generated at certain specified intervals.

The upper channel is a channel which generates a real-time output trigger (TAUDnTRC.TAUDnTRCm = 1), and the lower channel is a channel which makes a real-time output in response to the upper channel trigger (TAUDnTRC.TAUDnTRCm = 0).

#### Prerequisites

- Channels should use the TAUDTTOUTm control of other channels.
- The operating mode for the upper channel should be set to interval timer mode. (See **Table 25.118, Contents of the TAUDnCMORm Register for the Upper Channel of Real-Time Output Function Type 1**.)
- Any operating mode can be set for lower channels.
- The channel output mode for all the channels should be set to independent channel output mode 1 with real-time output. (See **Section 25.7, Channel Output Modes**.)
- Real-time output should be enabled for the upper channel (TAUDnTRE.TAUDnTREm = 1).

#### Functional description

The counter of the upper channel is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This in turn sets TAUDnTE.TAUDnTEm to 1, enabling count operation. The current value of the data register of the upper channel (TAUDnCDRm) is loaded into the counter (TAUDnCNTm) and the counter starts to count down from this value.

When the counter of the upper channel reaches $0000_H$, INTTAUDnIm is generated and TAUDTTOUTm outputs the current value of the real-time output bit (TAUDnTRO.TAUDnTROm) of every channel (only channels with TAUDnTRE.TAUDnTREm = 1). TAUDnCNTm then reloads the TAUDnCDRm value to continue operation subsequently.

The TAUDTTOUTm signal changes only when an interrupt is generated, and when its value is different from the current value of TAUDnTRO.TAUDnTROm at the moment that the interrupt is generated.

#### Conditions

- The channel which is monitored for INTTAUDnIm occurrence is specified by setting TAUDnTRC.TAUDnTRCm to 1 for the corresponding channel. The TAUDnTRC.TAUDnTRCm bit should be 0 for all other channels that do not generate a real-time output trigger.
- If real-time output of a lower channel is disabled (TAUDnTRE.TAUDnTREm = 0) or if the

channel itself is used as a rewrite trigger (TAUDnTRC.TAUDnTRCm = 1), the value of that channel's TAUDnTRO.TAUDnTROm bit is output when INTTAUDnIm is generated in that channel.

- If real-time output of a lower channel is enabled (TAUDnTRE.TAUDnTREm = 1) and TAUDnTRC.TAUDnTRCm = 0, the value of that channel's TAUDnTRO.TAUDnTROm bit is output when INTTAUDnIm is generated in the upper channel.
- If the TAUDnCMORm.TAUDnMD0 bit is set to 0, the first interrupt after a start or restart is not output. For details, see **Section 25.9, TAUDTTOUTm Output and INTTAUDnIm Generation when Counter Starts or Restarts**.

### 25.13.1.2 Equations

INTTAUDnIm generation cycle = count clock cycle × (TAUDnCDRm value + 1)

### 25.13.1.3 Block Diagram and General Timing Diagram

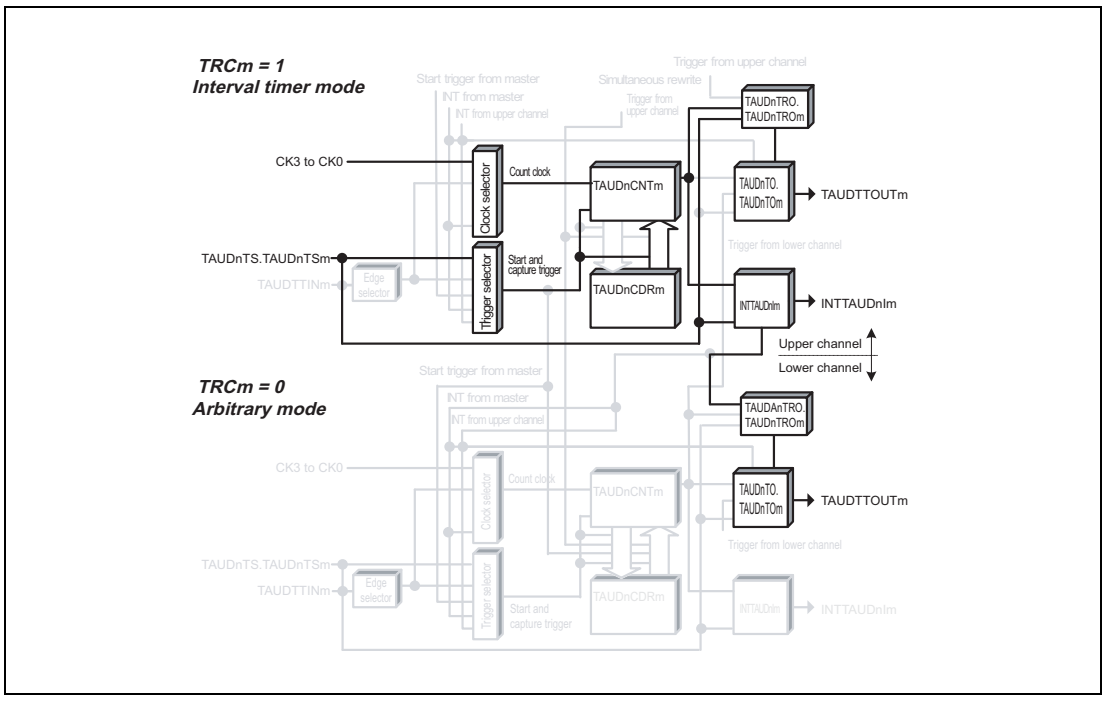



**Figure 25.80 Block Diagram of Real-Time Output Function Type 1**

The following settings apply to the general timing diagram.

- INTTAUDnIm is generated at the beginning of operation. (TAUDnCMORm.TAUDnMD0 = 1)

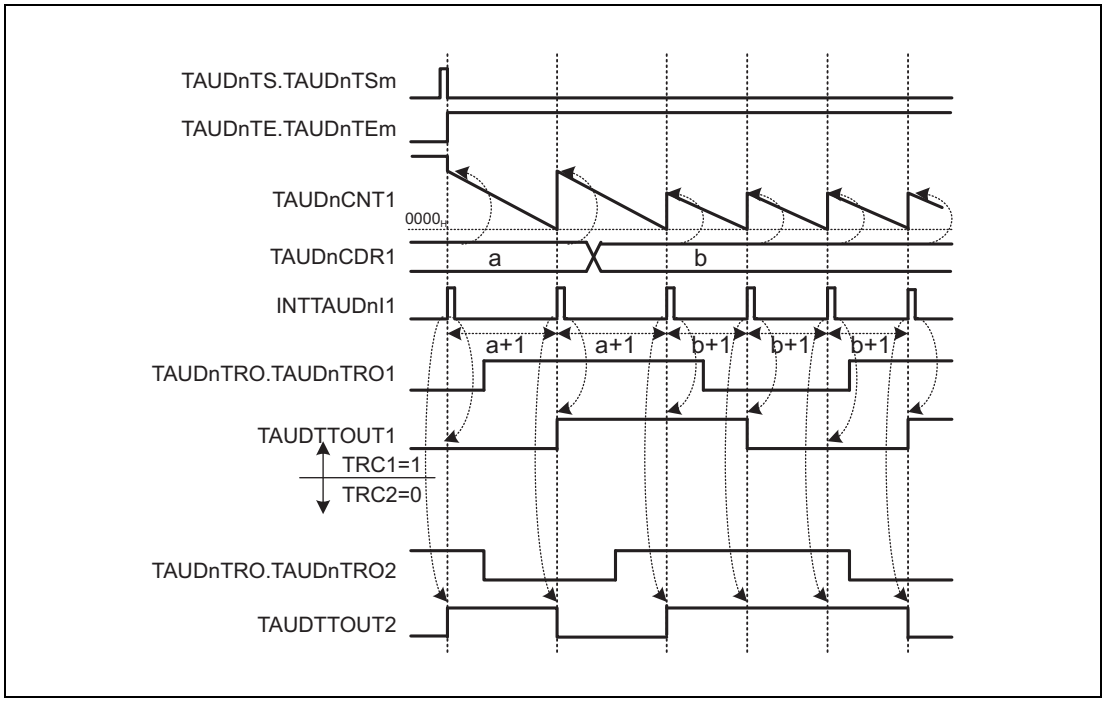



**Figure 25.81 General Timing Diagram of Real-Time Output Function Type 1**

### 25.13.1.4 Register Settings for Upper Channels

#### (1) TAUDnCMORm for upper channels

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.118 Contents of the TAUDnCMORm Register for the Upper Channel of Real-Time Output Function Type 1**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation, set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 000: Trigger the counter using software. |
| 7, 6 | TAUDnCOS [1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0000: Interval timer mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm not generated at the beginning of operation.<br>1: INTTAUDnIm generated at the beginning of operation. |

#### (2) TAUDnCMURm for upper channels

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.119 Contents of the TAUDnCMURm Register for the Upper Channel of Real-Time Output Function Type 1**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for upper channels

Table 25.120 Control Bit Settings in Independent Channel Output Mode 1 with Real-Time Output

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 0: Independent channel output |
| TAUDnTOC.TAUDnTOCm | 0: Operating mode 1 (Toggle mode if TAUDnTOM.TAUDnTOMm = 0) |
| TAUDnTOL.TAUDnTOLm | 0: The setting is disabled in toggle mode. (The value after reset.) |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 1: Enables real-time output |
| TAUDnTRO.TAUDnTROm | 0: Real-time output is low<br>1: Real-time output is high |
| TAUDnTRC.TAUDnTRCm | 1: Channel m generates a unique real-time output trigger |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

### (4) Simultaneous rewrite for upper channels

The simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the real-time output function type 1. Therefore, these registers should be set to 0.

Table 25.121 Simultaneous Rewrite Settings for Real-Time Output Function Type 1

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

#### 25.13.1.5 Register Settings for Lower Channels

**(1) TAUDnCMORm for lower channels**

The TAUDnCMORm register for lower channels can be set arbitrarily.

**(2) TAUDnCMURm for lower channels**

The TAUDnCMURm register for lower channels can be set arbitrarily.

**(3) Channel output mode for lower channels**

**Table 25.122 Control Bit Settings for the Lower Channels in Independent Channel Output Mode 1 with Real-Time Output**

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 0: Independent channel output |
| TAUDnTOC.TAUDnTOCm | 0: Operating mode 1 (Toggle mode if TAUDnTOM.TAUDnTOMm = 0) |
| TAUDnTOL.TAUDnTOLm | 0: The setting is disabled in toggle mode. (The value after reset.) |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set this bit to 0 |
| TAUDnTDL.TAUDnTDLm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set this bit to 0 |
| TAUDnTRE.TAUDnTREm | 1: Enables real-time output |
| TAUDnTRO.TAUDnTROm | 0: Real-time output is low<br>1: Real-time output is high |
| TAUDnTRC.TAUDnTRCm | 0: Upper channel generates a real-time output trigger for channel m |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

**(4) Simultaneous rewrite for lower channels**

Simultaneous rewrite registers for lower channels are available for any setting.

### 25.13.1.6 Operating Procedure for Real-Time Output Function Type 1

Table 25.123 Operating Procedure for Real-Time Output Function Type 1

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers for upper channels as described in **Table 25.118, Contents of the TAUDnCMORm Register for the Upper Channel of Real-Time Output Function Type 1**, and **Table 25.119, Contents of the TAUDnCMURm Register for the Upper Channel of Real-Time Output Function Type 1**.<br><br>Set TAUDnCMORm and TAUDnCMURm registers for lower channels as described in **Section 25.13.1.5, Register Settings for Lower Channels**.<br><br>Set the value of TAUDnCDRm register (only channels with TAUDnTRC.TAUDnTRCm = 1)<br><br>Set channel output mode by setting the control bits as described in **Table 25.120, Control Bit Settings in Independent Channel Output Mode 1 with Real-Time Output**.<br><br>Set channel output mode by setting the control bits as described in **Table 25.122, Control Bit Settings for the Lower Channels in Independent Channel Output Mode 1 with Real-Time Output**. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm = 1 on the channel with TAUDnTRC.TAUDnTRCm set to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | [Channel with TAUDnTRC.TAUDnTRCm set to 1]<br>TAUDnTE.TAUDnTEm is set to 1 and the counter starts.<br>The TAUDnCDRm value is loaded in TAUDnCNTm. If TAUDnCMORm.TAUDnMD0 is 1, INTTAUDnIm is generated. |
| During Operation | TAUDnCDRm and TAUDnTRO.TAUDnTROm can be changed at any time.<br>The TAUDnCNTm register can be read at any time. | TAUDnCNTm counts down. When the counter reaches $0000_H$:<br>• The TAUDnCDRm value is loaded in TAUDnCNTm again and count operation continues.<br>• INTTAUDnIm is generated.<br>• TAUDTTOUTm outputs the current value of the real-time output bit TAUDnTRO.TAUDnTROm.<br>Afterwards, this procedure is repeated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops. Both TAUDnCNTm and TAUDTTOUTm retain their current values. |

Restart Operation

### 25.13.1.7 Specific Timing Diagrams

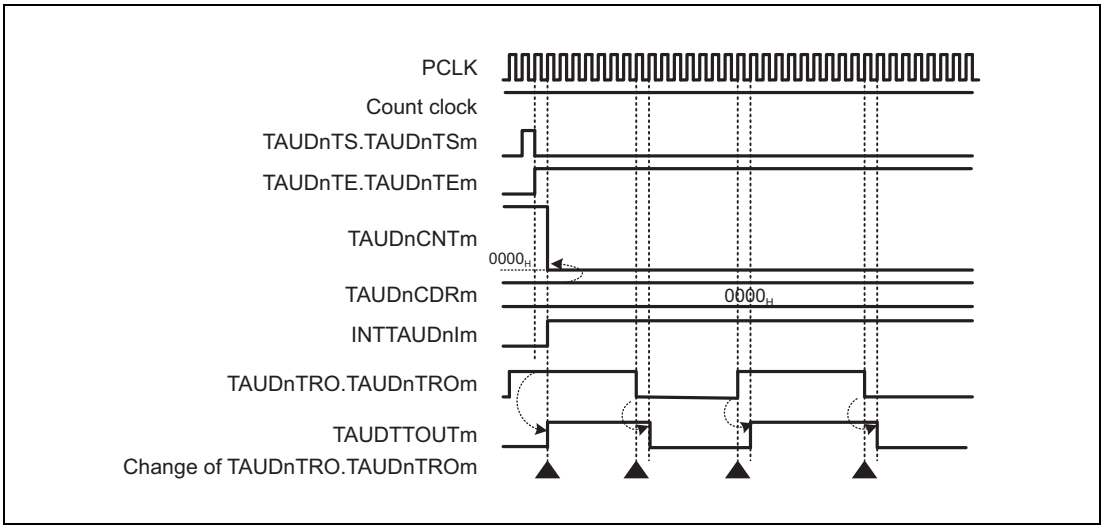



**Figure 25.82 TAUDnCDRm = $0000_H$, TAUDnCMORm.TAUDnMD0 = 1**

- The value of TAUDTTOUTm changes according to the setting of TAUDnTRO.TAUDnTROm with a delay of one PCLK cycle.

## 25.13.2 Real-Time Output Function Type 2

### 25.13.2.1 Overview

#### Summary

This function outputs the value of TAUDnTRO.TAUDnTROm bit from TAUDTTOUTm when a specified channel generates an interrupt (INTTAUDnIm). In this function, the interrupt is generated when a valid TAUDTTINm input edge is detected or the function starts.

The upper channel is a channel which generates a real-time output trigger (TAUDnTRC.TAUDnTRCm = 1), and the lower channel is a channel which makes a real-time output in response to the upper channel trigger (TAUDnTRC.TAUDnTRCm = 0).

#### Prerequisites

- Channels should use the TAUDTTOUTm control of the other channels.
- The operating mode for the upper channel should be set to capture mode. (See **Table 25.124, Contents of the TAUDnCMORm Register for the Upper Channel of Real-Time Output Function Type 2**.)
- Any operating mode can be set for lower channels.
- The channel output mode for all the channels should be set to independent channel output mode 1 with real-time output. (See **Section 25.7, Channel Output Modes**.)
- Real-time output should be enabled for the upper channel (TAUDnTRE.TAUDnTREm = 1).

#### Functional description

The counter for upper channels is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This sets TAUDnTE.TAUDnTEm to 1, enabling count operation. The counter starts to count up.

When a valid TAUDTTINm input edge is generated on one of upper channels, an interrupt occurs and TAUDTTOUTm outputs the current value of the real-time output bit (TAUDnTRO.TAUDnTROm) of every channel (only channels with TAUDnTRE.TAUDnTREm = 1).

The TAUDTTOUTm signal changes only when an interrupt is generated, and when TAUDTTOUTm value is different from the current value of TAUDnTRO.TAUDnTROm during the occurrence of the interrupt.

#### Conditions

- The channel which is monitored for INTTAUDnIm occurrence is specified by setting TAUDnTRC.TAUDnTRCm to 1 for the corresponding channel. The TAUDnTRC.TAUDnTRCm bit should be 0 for all other channels that do not generate a real-time output trigger.
- If real-time output of a lower channel is disabled (TAUDnTRE.TAUDnTREm = 0) or if the channel itself is used as a rewrite trigger (TAUDnTRC.TAUDnTRCm = 1), the value of that channel's TAUDnTRO.TAUDnTROm bit is output when INTTAUDnIm is generated in that channel.
- If real-time output of a lower channel is enabled (TAUDnTRE.TAUDnTREm = 1) and TAUDnTRC.TAUDnTRCm = 0, the value of that channel's TAUDnTRO.TAUDnTROm bit is output when INTTAUDnIm is generated in the upper channel.
- If the TAUDnCMORm.TAUDnMD0 bit is set to 0, the first interrupt after a start or restart is not

output. For details, see **Section 25.9, TAUDTTOUTm Output and INTTAUDnIm Generation when Counter Starts or Restarts**.

### 25.13.2.2 Block Diagram and General Timing Diagram

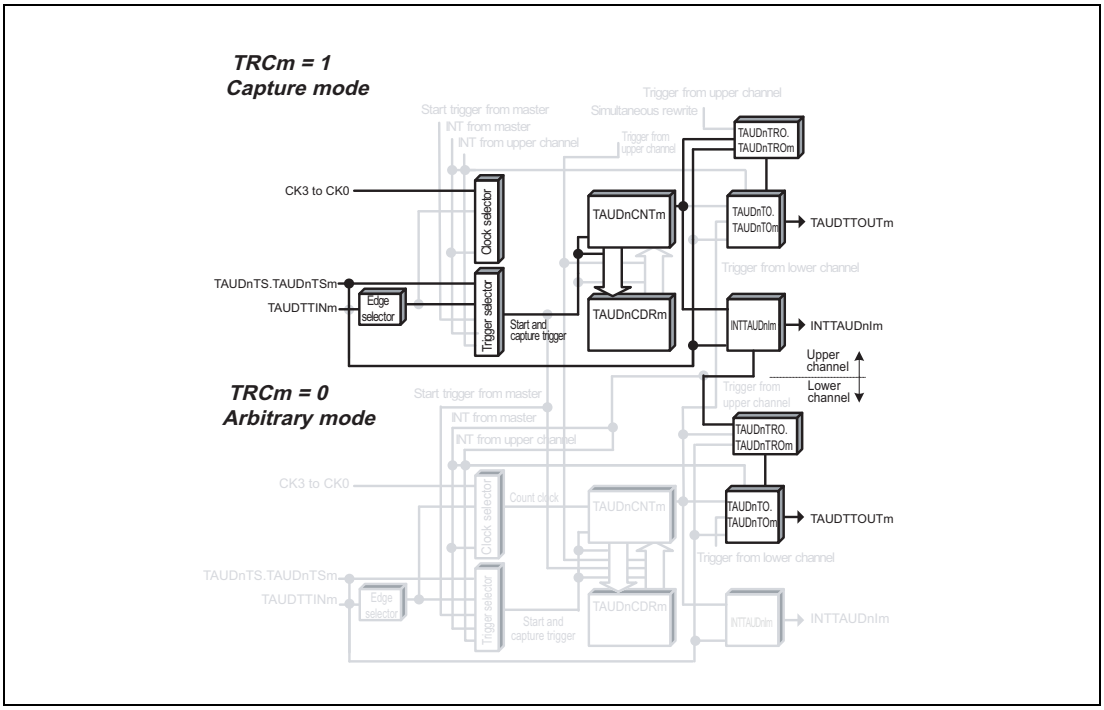



**Figure 25.83 Block Diagram of Real-Time Output Function Type 2**

The following settings apply to the general timing diagram.

- INTTAUDnIm is not generated at the beginning of operation. (TAUDnCMORm.TAUDnMD0 = 0)

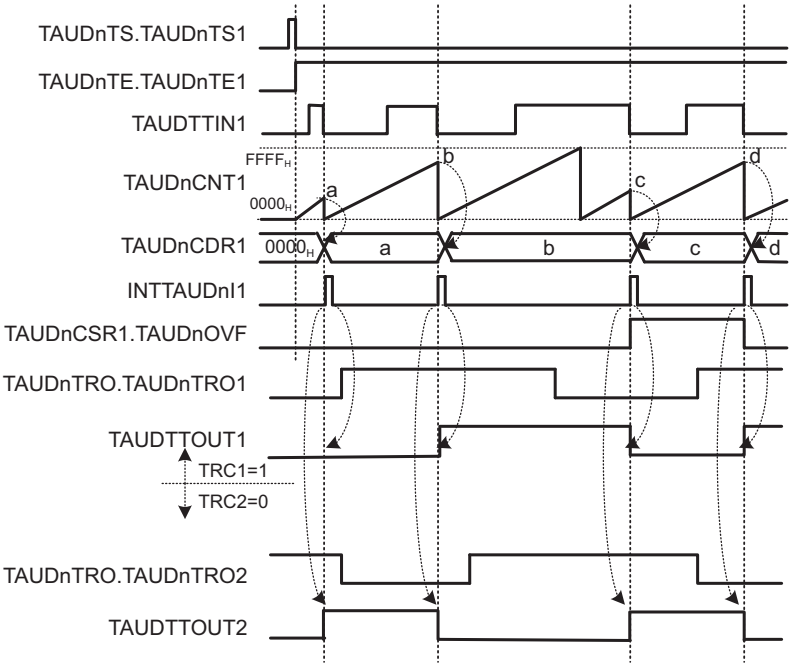



**Figure 25.84 General Timing Diagram of Real-Time Output Function Type 2**

### 25.13.2.3 Register Settings for Upper Channels

#### (1) TAUDnCMORm for upper channels

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.124 Contents of the TAUDnCMORm Register for the Upper Channel of Real-Time Output Function Type 2**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation, set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 001: Valid edge of the TAUDTTINm input signal is used as an external start trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0010: Capture mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm not generated at the beginning of operation.<br>1: INTTAUDnIm generated at the beginning of operation. |

#### (2) TAUDnCMURm for upper channels

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.125 Contents of the TAUDnCMURm Register for the Upper Channel of Real-Time Output Function Type 2**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Detection of falling edge<br>01: Detection of rising edge<br>10: Detection of rising and falling edges<br>11: Setting prohibited |

### (3) Channel output mode for upper channels

**Table 25.126 Control Bit Settings in Independent Channel Output Mode 1 with Real-Time Output**

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 0: Independent channel output |
| TAUDnTOC.TAUDnTOCm | 0: Operating mode 1 (Toggle mode if TAUDnTOM.TAUDnTOMm = 0) |
| TAUDnTOL.TAUDnTOLm | 0: The setting is disabled in toggle mode. (The value after reset.) |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 1: Enables real-time output |
| TAUDnTRO.TAUDnTROm | 0: Real-time output is low<br>1: Real-time output is high |
| TAUDnTRC.TAUDnTRCm | 1: Channel m generates a unique real-time output trigger |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

### (4) Simultaneous rewrite for upper channels

The simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the real-time output function type 2. Therefore, these registers should be set to 0.

**Table 25.127 Simultaneous Rewrite Settings for Real-Time Output Function Type 2**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

### 25.13.2.4 Register Settings for Lower Channels

#### (1) TAUDnCMORm for lower channels

The TAUDnCMORm register for lower channels can be set arbitrarily.

#### (2) TAUDnCMURm for lower channels

The TAUDnCMURm register for lower channels can be set arbitrarily.

#### (3) Channel output mode for lower channels

**Table 25.128 Control Bit Settings for Lower Channels in Independent Channel Output Mode 1 with Real-Time Output**

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 0: Independent channel output |
| TAUDnTOC.TAUDnTOCm | 0: Operating mode 1 (Toggle mode if TAUDnTOM.TAUDnTOMm = 0) |
| TAUDnTOL.TAUDnTOLm | 0: The setting is disabled in toggle mode. (The value after reset.) |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 1: Enables real-time output. |
| TAUDnTRO.TAUDnTROm | 0: Real-time output is low<br>1: Real-time output is high |
| TAUDnTRC.TAUDnTRCm | 0: Upper channel generates a real-time output trigger for channel m |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

#### (4) Simultaneous rewrite for lower channels

Simultaneous rewrite registers for lower channels can be set arbitrarily.

### 25.13.2.5 Operating Procedure for Real-Time Output Function Type 2

Table 25.129 Operating Procedure for Real-Time Output Function Type 2

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers for upper channels as described in **Table 25.124, Contents of the TAUDnCMORm Register for the Upper Channel of Real-Time Output Function Type 2**, and **Table 25.125, Contents of the TAUDnCMURm Register for the Upper Channel of Real-Time Output Function Type 2**.<br><br>Set TAUDnCMORm and TAUDnCMURm registers for the lower channel as described in **Section 25.13.2.4, Register Settings for Lower Channels**.<br><br>The TAUDnCDRm register functions as a capture register (only channels with TAUDnTRC.TAUnTRCm = 1).<br><br>Set channel output mode by setting the control bits as described in **Table 25.126, Control Bit Settings in Independent Channel Output Mode 1 with Real-Time Output**.<br><br>Set channel output mode by setting the control bits as described in **Table 25.128, Control Bit Settings for Lower Channels in Independent Channel Output Mode 1 with Real-Time Output**. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm = 1 on the channel with TAUDnTRC.TAUDnTRCm set to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | [Channel with TAUDnTRC.TAUDnTRCm set to 1]<br>TAUDnTE.TAUDnTEm is set to 1 and the counter starts.<br>TAUDnCNTm is cleared to $0000_H$. If TAUDnCMORm.TAUDnMD0 is 1, INTTAUDnIm is generated. |
| During Operation | TAUDnTRO.TAUDnTROm can be changed at any time. | TAUDnCNTm starts to count up from $0000_H$.<br>When a valid TAUDTTINm input edge is detected:<br>• TAUDnCNTm captures the TAUDnCDRm value, and the counter is cleared to $0000_H$.<br>• INTTAUDnIm is generated.<br>• When the TAUDTTINm input valid edge is detected immediately after the generation of an overflow, the TAUDnCSRm.TAUDnOVF bit is set to 1. When detected before the generation of an overflow, the TAUDnCSRm.TAUDnOVF bit is cleared to 0.<br>TAUDTTOUTm outputs the current value of real-time output bit TAUDnTRO.TAUDnTROm.<br>Afterwards, this procedure is repeated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops. TAUDnCNTm, TAUDnCSRm.TAUDnOVF, and TAUDTTOUTm retain their current values. |

Restart Operation

### 25.13.2.6 Specific Timing Diagrams

#### (1) Operation start and stop

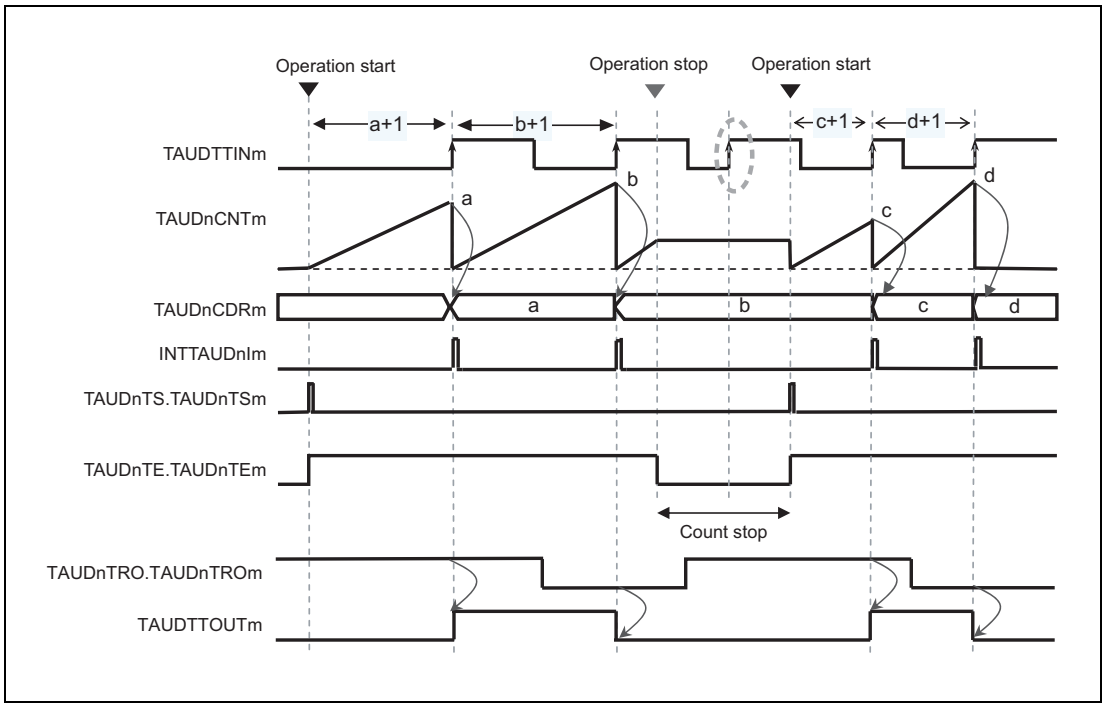



**Figure 25.85 Operation Start and Stop (TAUDnCMORm.TAUDnMD0 = 0)**

- When TAUDnTS.TAUDnTSm is set to 1, the counter starts counting up.
- When a valid input edge is detected, the current value of the counter is written to the data register (TAUDnCDRm) and an interrupt is generated.
- TAUDTTOUTm outputs the current value of the real-time output bit (TAUDnTRO.TAUDnTROm) and the counter resets and starts to count up again.
- The TAUDTTOUTm signal only changes when an interrupt is generated, and then only when its value is different from the current value of TAUDnTRO.TAUDnTROm at the moment that the interrupt is generated.
- If the counter is stopped (TAUDnTE.TAUDnTEm = 0), valid input edges are ignored and no interrupt is generated.

# 25.14 Independent Channel Simultaneous Rewrite Functions

This section describes functions that carry out simultaneous rewrite.

## 25.14.1 Simultaneous Rewrite Trigger Generation Function Type 1

### 25.14.1.1 Overview

#### Summary

This function generates an interrupt on a specific channel that can be used by lower channels as a simultaneous rewrite trigger. The interrupt is generated at regular intervals.

The upper channel is a channel which generates a simultaneous rewrite trigger (TAUDnRDC.TAUDnRDCm = 1), and the lower channel is a channel which makes a simultaneous rewrite in response to the upper channel trigger (TAUDnRDC.TAUDnRDCm = 0).

#### Prerequisites

- Two or more channels lower than the channel used as upper channel are enabled for simultaneous rewrite (TAUDnRDE.TAUDnRDEm = 1).
- The operating mode for the upper channel should be set to interval timer mode. (See **Table 25.130, Contents of the TAUDnCMORm Register for the Upper Channel of Simultaneous Rewrite Trigger Generation Function Type 1**.)
- For the operating mode that can be set for lower channels, see **Table 25.47, Channel Functions and the Methods They Use for Simultaneous Rewrite**.
- TAUDTTOUTm is not used for any channel in this function.

#### Functional description

The counter operation is enabled by setting the channel trigger bits (TAUDnTS.TAUDnTSm) for upper and lower channels to 1. This sets TAUDnTE.TAUDnTEm = 1, enabling count operation. The current value of the data register buffer for upper channels (TAUDnCDRm buf) is loaded into the counter (TAUDnCNTm) and the counter starts to count down from this value. The counter for lower channels start to count according to the selected operating mode.

Once the counter reaches $0000_H$, an interrupt occurs on the channel. The current value of the corresponding TAUDnCDRm buffer is loaded into TAUDnCNTm to continue operation subsequently.

If the channel where an interrupt occurs is specified as a trigger channel for simultaneous rewrite (TAUDnRDC.TAUDnRDCm = 1) and is an upper channel, simultaneous rewrite takes place on all lower channels in which simultaneous rewrite is currently possible (TAUDnRSF.TAUDnRSFm = 1).

The values of the data registers are copied to the corresponding data register buffers. Each time a counter starts to count down, it reads the value in the data register buffer and counts down from this value.

The value of a data register can be changed at any time, but it is only transferred to the corresponding data register buffer when simultaneous rewrite occurs.

#### Conditions

- The channel which is monitored for INTTAUDnIm occurrence is specified by setting TAUDnRDC.TAUDnRDCm = 1 for the corresponding channel. The TAUDnRDC.TAUDnRDCm

bit should be set to 0 for all other channels in which simultaneous rewrite should take place.

- If the TAUDnCMORm.TAUDnMD0 bit is set to 0, the first interrupt after a start or restart is not generated. For details, see **Section 25.9, TAUDTTOUTm Output and INTTAUDnIm Generation when Counter Starts or Restarts**.

#### 25.14.1.2 Equations

Simultaneous rewrite trigger generation cycle = count clock cycle × (TAUDnCDRm + 1)

To control simultaneous rewrite, the following condition should be satisfied:

[PWM]

TAUDnCDRm = [(value of TAUDnCDRm of master channel subject to simultaneous rewrite + 1) × number of interrupts] – 1

[Triangle PWM]

TAUDnCDRm = [(value of TAUDnCDRm of master channel subject to simultaneous rewrite + 1) × 2 × number of interrupts] – 1

That is, the ratio of TAUDnCDRm + 1 and value of TAUDnCDRm of master channel subject to simultaneous rewrite + 1 must be an integer. This integer corresponds to the number of interrupts.

For triangle PWM, remember that the cycle doubles.

### 25.14.1.3 Block Diagram and General Timing Diagram

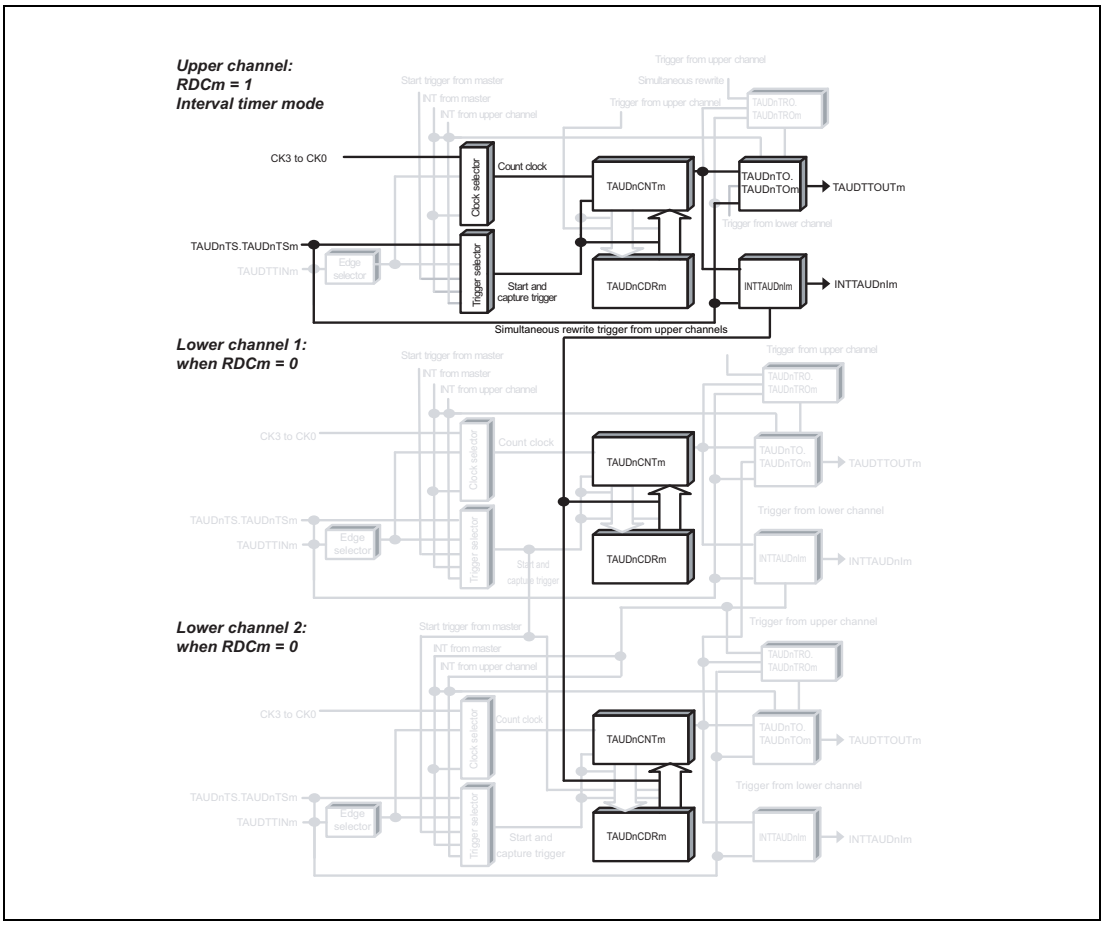



**Figure 25.86 Block Diagram of Simultaneous Rewrite Trigger Generation Function Type 1**

The following settings apply to the general timing diagram.

- INTTAUDnIm is generated at the beginning of operation. (TAUDnCMORm.TAUDnMD0 = 1)

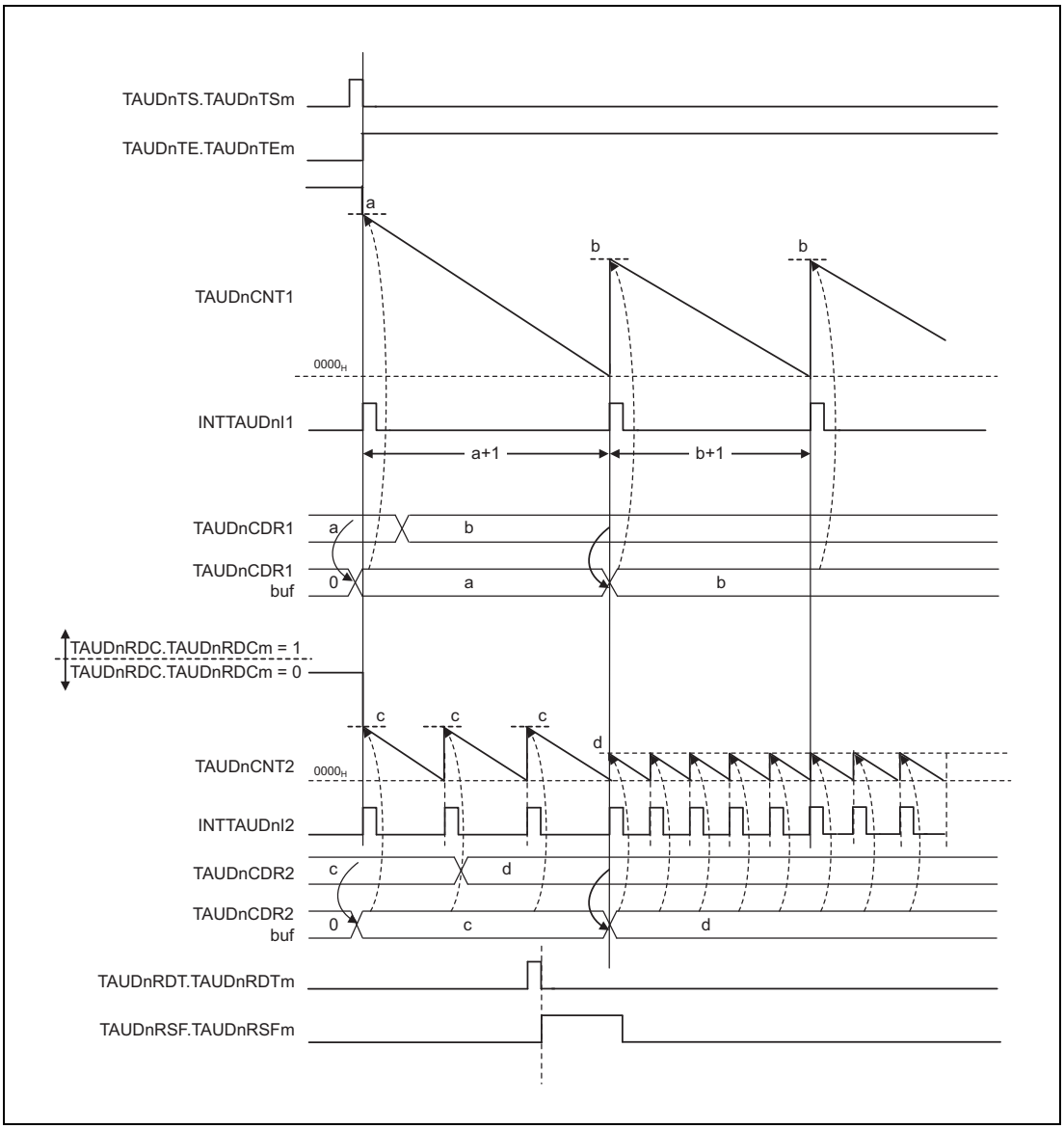



**Figure 25.87 General Timing Diagram of Simultaneous Rewrite Trigger Generation Function Type 1**

### 25.14.1.4 Register Settings for Upper Channels

#### (1) TAUDnCMORm for upper channels

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.130 Contents of the TAUDnCMORm Register for the Upper Channel of Simultaneous Rewrite Trigger Generation Function Type 1**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation. Set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 000: Trigger the counter using software. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0000: Interval timer mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm not generated at the beginning of operation.<br>1: INTTAUDnIm generated at the beginning of operation. |

#### (2) TAUDnCMURm for upper channels

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.131 Contents of the TAUDnCMURm Register for the Upper Channel of Simultaneous Rewrite Trigger Generation Function Type 1**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

#### (3) Channel output mode for upper channels

TAUDnTOE.TAUDnTOEm is set to 0 because the channel output mode is not used with this function. However, this mode can be used in independent channel output mode controlled by software.

#### (4) Simultaneous rewrite for upper channels

Table 25.132 Simultaneous Rewrite Settings for Upper Channels in Simultaneous Rewrite Trigger Generation Function Type 1

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 1: Selects one of upper channels as simultaneous rewrite control channel. |
| TAUDnRDM.TAUDnRDMm | 0: Loads a simultaneous rewrite control signal when the master channel starts counting. |
| TAUDnRDC.TAUDnRDCm | 1: Monitors INTTAUDnIm signal which triggers a simultaneous rewrite on the channel. |

### 25.14.1.5 Register Settings for Lower Channels

#### (1) TAUDnCMORm for lower channels

TAUDnCMORm register for lower channels must follow the TAUDnCMORm register settings in the operating mode which can be set. (See **Table 25.47, Channel Functions and the Methods They Use for Simultaneous Rewrite**.)

#### (2) TAUDnCMURm for lower channels

TAUDnCMURm register for lower channels must follow the TAUDnCMURm register settings in the operating mode which can be set. (See **Table 25.47, Channel Functions and the Methods They Use for Simultaneous Rewrite**.)

#### (3) Channel output mode for lower channels

Output can be made according to the setting for lower channels (master/slave). As for the available function for simultaneous rewrite trigger generation function type 1, see **Table 25.46, Simultaneous Rewrite Methods and when They are Triggered**.

#### (4) Simultaneous rewrite for lower channels

Table 25.133 Simultaneous Rewrite Settings for Lower Channels in Simultaneous Rewrite Trigger Generation Function Type 1

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 1: Selects one of upper channels as simultaneous rewrite control channel. |
| TAUDnRDM.TAUDnRDMm | 0: Loads a simultaneous rewrite control signal when the master channel starts counting. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

### 25.14.1.6 Operating Procedure for Simultaneous Rewrite Trigger Generation Function Type 1

Table 25.134 Operating Procedure for Simultaneous Rewrite Trigger Generation Function Type 1

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set TAUDnCMORm and TAUDnCMURm registers for the upper channel as described in **Table 25.130, Contents of the TAUDnCMORm Register for the Upper Channel of Simultaneous Rewrite Trigger Generation Function Type 1**, and **Table 25.131, Contents of the TAUDnCMURm Register for the Upper Channel of Simultaneous Rewrite Trigger Generation Function Type 1**.<br><br>Set TAUDnCMORm and TAUDnCMURm registers for lower channels as described in **Section 25.14.1.5, Register Settings for Lower Channels**.<br><br>Set the value of TAUDnCDRm register. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is set to 1 and the counter starts.<br>TAUDnCDRm value is loaded into TAUDnCNTm. If TAUDnCMORm.TAUDnMD0 = 1, INTTAUDnIm is generated. |
| During Operation | TAUDnRDT.TAUDnRDTm and TAUDnCDRm.TAUDnCDR are changeable.<br>TAUDnRSF.TAUDnRSFm can be always read. | TAUDnCNTm counts down. When the counter reaches $0000_H$:<br>• The TAUDnCDRm value is loaded in TAUDnCNTm again and count operation continues.<br>• INTTAUDnIm is generated.<br>If INTTAUDnIm is generated on the channel where TAUDnRDC.TAUDnRDCm is set to 1, simultaneous rewrite is controlled.<br>Afterwards, this procedure is repeated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops and retains its current value. |

Restart Operation

## 25.14.2 Simultaneous Rewrite Trigger Generation Function Type 2

### 25.14.2.1 Overview

#### Summary

This function generates an interrupt on a specific channel that can be used by lower channels as a simultaneous rewrite trigger. In this function, the interrupt is generated when a valid TAUDTTINm input edge is detected or the function starts.

The upper channel is a channel which generates a simultaneous rewrite trigger (TAUDnRDC.TAUDnRDCm = 1), and the lower channel is a channel which makes a simultaneous rewrite in response to the upper channel trigger (TAUDnRDC.TAUDnRDCm = 0).

#### Prerequisites

- Two or more channels lower than the channel used as upper channel are enabled for simultaneous rewrite (TAUDnRDE.TAUDnRDEm = 1).
- The operation mode of the upper channel must be set to Capture Mode (see **Table 25.135, Contents of the TAUDnCMORm Register for the Upper Channel of Simultaneous Rewrite Trigger Generation Function Type 2**).
- For the operation mode that can be set for a lower channel, see **Table 25.47, Channel Functions and the Methods They Use for Simultaneous Rewrite**.

#### Functional description

The counter operation is enabled by setting the channel trigger bits (TAUDnTS.TAUDnTSm) for upper and lower channels to 1. This sets TAUDnTE.TAUDnTEm = 1, enabling count operation. The counter for the upper channel starts to count up, and then the counter for lower channels start to count according to the selected operating mode.

When a TAUDTTINm input edge occurs on the upper channel, an interrupt is generated. The trigger is detected by the lower channel(s), which then also generate an interrupt.

When TAUDnRDC.TAUDnRDCm = 1 on the upper channel, simultaneous rewrite takes place on all lower channels in which simultaneous rewrite is currently possible (TAUDnRSF.TAUDnRSFm = 1).

The values of the data registers are copied to the corresponding data register buffers.

The value of a data register can be changed at any time, but it is only transferred to the corresponding data register buffer when simultaneous rewrite occurs.

#### Conditions

- The channel which is monitored for INTTAUDnIm is specified by setting TAUDnRDC.TAUDnRDCm = 1 for the corresponding channel. The TAUDnRDC.TAUDnRDCm bit must be 0 for all other channels in which simultaneous rewrite should take place.
- If the TAUDnCMORm.TAUDnMD0 bit is set to 1, an interrupt is generated when the function starts. For details see **Section 25.9, TAUDTTOUTm Output and INTTAUDnIm Generation when Counter Starts or Restarts**.

### 25.14.2.2 Block Diagram and General Timing Diagram

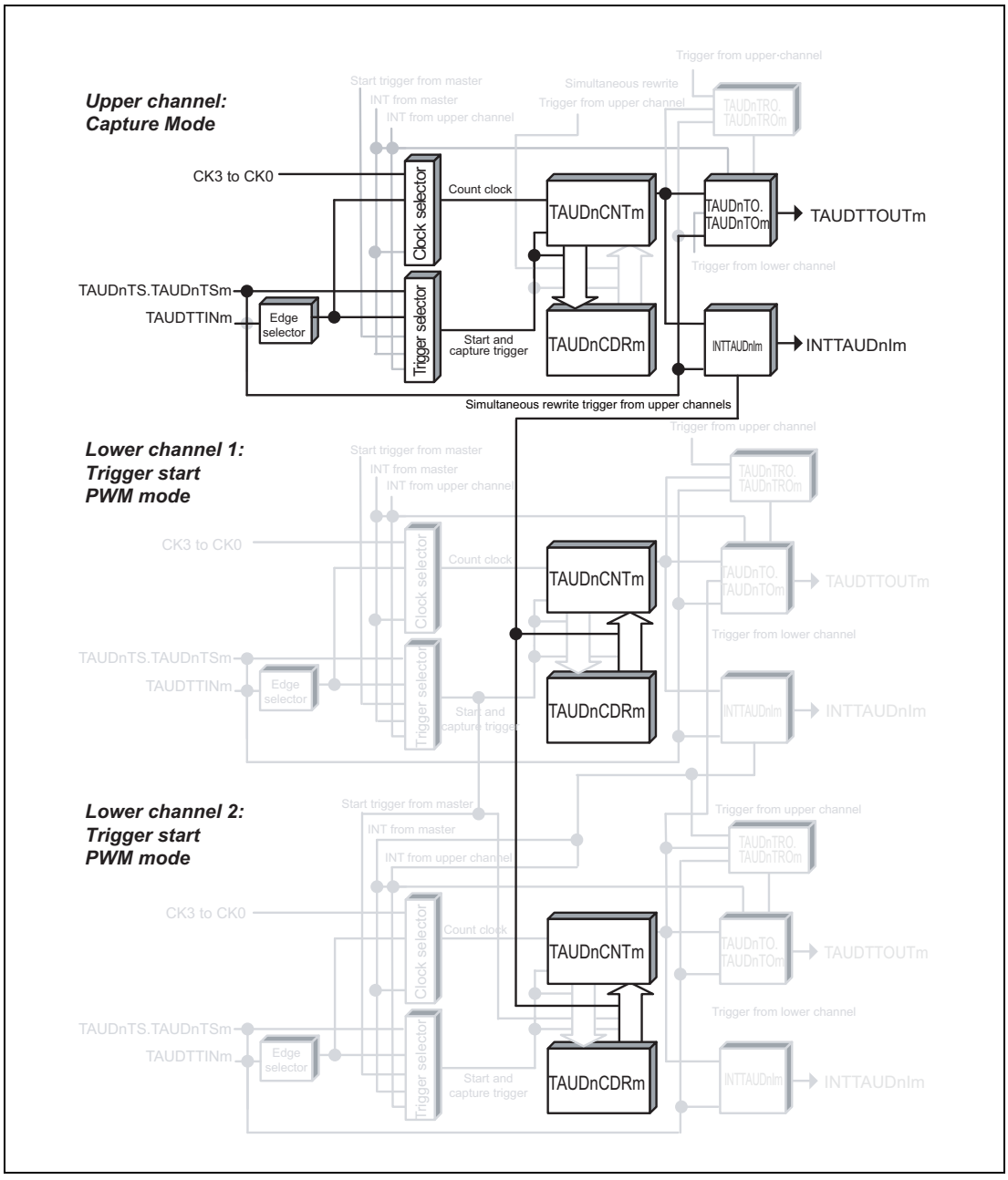



**Figure 25.88 Block Diagram for Simultaneous Rewrite Trigger Generation Function Type 2**

The following settings apply to the general timing diagram.

- INTTAUDnIm is generated at the beginning of operation. (TAUDnCMORm.TAUDnMD0 = 1)
- Detection of falling edge (TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)
- Upper channel (CH1) generates simultaneous rewrite trigger.

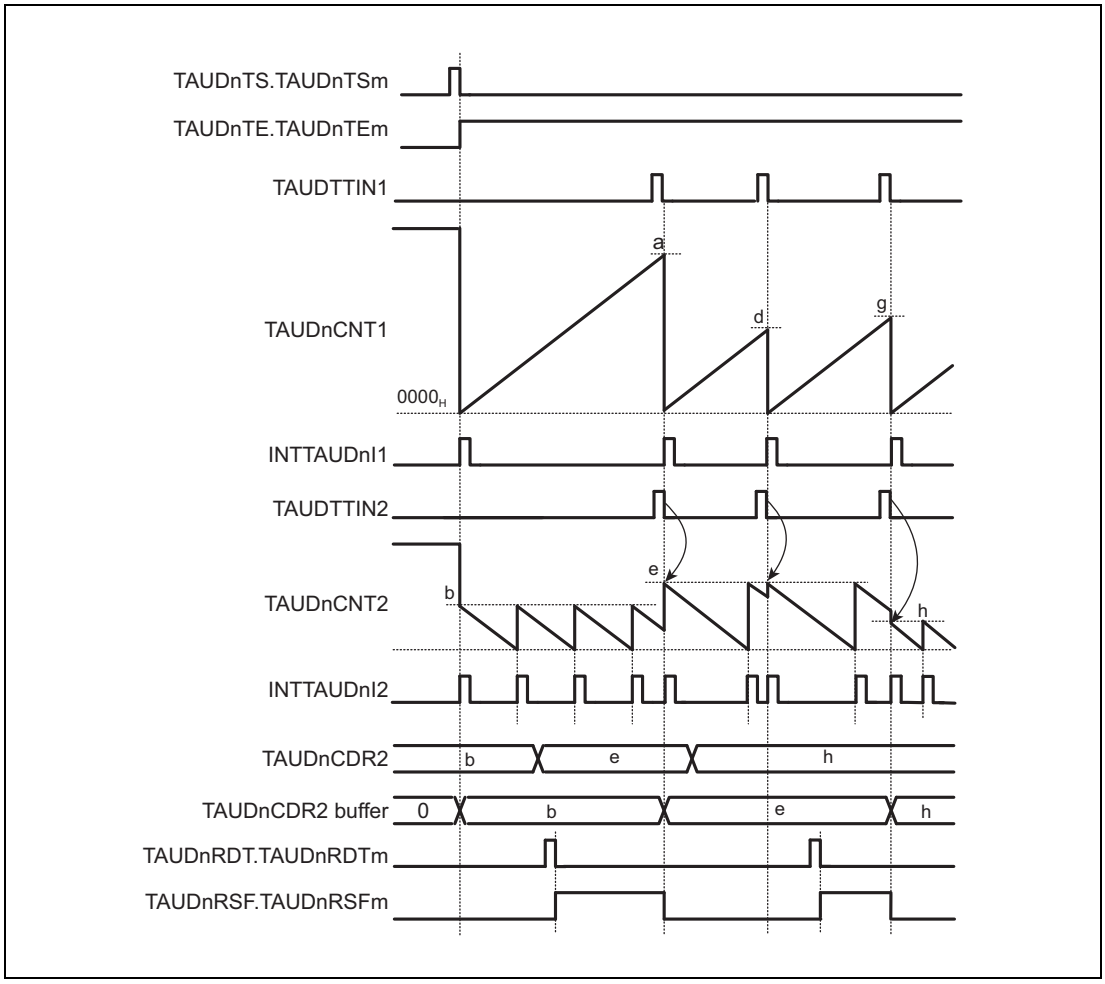



**Figure 25.89 General Timing Diagram for Simultaneous Rewrite Trigger Generation Function Type 2**

### 25.14.2.3 Register Settings for Upper Channels

#### (1) TAUDnCMORm for upper channels

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.135 Contents of the TAUDnCMORm Register for the Upper Channel of Simultaneous Rewrite Trigger Generation Function Type 2**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Independent operation, set to 0. |
| 10 to 8 | TAUDnSTS[2:0] | 001: Valid edge of the TAUDTTINm input signal is used as the external capture trigger |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0010: Capture mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm not generated at the beginning of operation.<br>1: INTTAUDnIm generated at the beginning of operation. |

#### (2) TAUDnCMURm for upper channels

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.136 Contents of the TAUDnCMURm Register for the Upper Channel of Simultaneous Rewrite Trigger Generation Function Type 2**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Detection of falling edge<br>01: Detection of rising edge<br>10: Detection of rising and falling edges<br>11: Setting prohibited |

### (3) Channel output mode for upper channels

The channel output mode is not used by this function.

### (4) Simultaneous rewrite for upper channels

**Table 25.137 Simultaneous Rewrite Settings for Simultaneous Rewrite Trigger Generation Function Type 2**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 1: Selects one of upper channels as simultaneous rewrite control channel. |
| TAUDnRDM.TAUDnRDMm | 0: Loads a simultaneous rewrite control signal when the master channel starts to count. |
| TAUDnRDC.TAUDnRDCm | 1: Monitors INTTAUDnIm signal which triggers a simultaneous rewrite on the channel. |

### 25.14.2.4 Register Settings for Lower Channels

#### (1) TAUDnCMORm for lower channels

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.138 Contents of the TAUDnCMORm Register for the Lower channel of Simultaneous Rewrite Trigger Generation Function Type 2**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 1: Master channel |
| 10 to 8 | TAUDnSTS[2:0] | 001: Valid TAUDTTINm input edge signal is used as the start trigger |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0000: Interval timer mode |
| 0 | TAUDnMD0 | 1: INTTAUDnIm generated at the beginning of operation. |

#### (2) TAUDnCMURm for lower channels

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.139 Contents of the TAUDnCMURm Register for the Lower Channel of Simultaneous Rewrite Trigger Generation Function Type 2**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Detection of falling edge<br>01: Detection of rising edge<br>10: Detection of rising and falling edges<br>11: Setting prohibited |

### (3) Channel output mode for lower channels

Output can be made according to the trigger start PWM mode setting.

### (4) Simultaneous rewrite for lower channels

**Table 25.140 Simultaneous Rewrite Settings for the Lower Channel in Simultaneous Rewrite Trigger Generation Function Type 2**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 1: Selects one of upper channels as simultaneous rewrite control channel. |
| TAUDnRDM.TAUDnRDMm | 0: Loads a simultaneous rewrite control signal when the master channel starts to count. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

### 25.14.2.5 Operating Procedure for Simultaneous Rewrite Trigger Generation Function Type 2

**Table 25.141 Operating Procedure for Simultaneous Rewrite Trigger Generation Function Type 2**

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Set the TAUDnCMORm and TAUDnCMURm registers for the upper channel as described in **Table 25.135, Contents of the TAUDnCMORm Register for the Upper Channel of Simultaneous Rewrite Trigger Generation Function Type 2** and **Table 25.136, Contents of the TAUDnCMURm Register for the Upper Channel of Simultaneous Rewrite Trigger Generation Function Type 2**.<br><br>Set the TAUDnCMORm and TAUDnCMURm registers for the lower channel as described in **Table 25.138, Contents of the TAUDnCMORm Register for the Lower channel of Simultaneous Rewrite Trigger Generation Function Type 2** and **Table 25.139, Contents of the TAUDnCMURm Register for the Lower Channel of Simultaneous Rewrite Trigger Generation Function Type 2**.<br><br>The TAUDnCDRm register functions as a capture register. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm to 1.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is set to 1 and the counter starts.<br>TAUDnCNTm is cleared to $0000_H$.<br>INTTAUDnIm is generated when TAUDnCMORm.TAUDnMD0 is set to 1. |
| During Operation | TAUDnRDT.TAUDnRDTm can be set at any time.<br>TAUDnRSF.TAUDnRSFm can be read at any time. | TAUDnCNTm counts up from $0000_H$. When a TAUDTTINm valid edge is detected:<br>• TAUDnCNTm transfers (captures) its value to TAUDnCDRm and returns to $0000_H$.<br>• INTTAUDnIm is generated.<br>Simultaneous rewrite is controlled when INTTAUDnIm is generated from the channel where TAUDnRDC.TAUDnRDCm is set to 1. Afterwards, this procedure is repeated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm to 1.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops and it retains its current value. |

Restart Operation

# 25.15 Synchronous Channel Operation Functions

This section lists all the synchronous channel operation functions provided by the timer array unit D. For a general overview of synchronous channel operation, see **Section 25.2, Overview**

This section describes functions that generate PWM signals at regular intervals.

## 25.15.1 PWM Output Function

### 25.15.1.1 Overview

#### Summary

This function generates multiple PWM outputs by using a master and multiple slave channels. It enables the pulse cycle (frequency) and the duty cycle of the TAUDTTOUTm to be set. The pulse cycle is set in the master channel. The duty cycle is set in the slave channel.

#### Prerequisites

- Two channels
- The operating mode for the master channel should be set to interval timer mode. (See **Table 25.142, Contents of the TAUDnCMORm Register for the Master Channel of the PWM Output Function**.)
- The operating mode for the slave channels should be set to one-count mode. (See **Table 25.145, Contents of the TAUDnCMORm Register for the Slave Channel of the PWM Output Function**.)
- TAUDTTOUTm is not used with the master channel of this function.
- The channel output mode for the slave channels should be set to Synchronous Channel Output Mode 1. (See **Section 25.7, Channel Output Modes**.)

#### Functional description

The counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This sets TAUDnTE.TAUDnTEm = 1, enabling count operation. The current value of TAUDnCDRm is loaded into TAUDnCNTm, and the counter starts counting down from the TAUDnCDRm value. If an INTTAUDnIm is generated on the master channel and TAUDTTOUTm (slave) is set/reset, PWM output is made.

- Master channel:
  When the master channel counter reaches $0000_H$ and the pulse cycle time has passed, INTTAUDnIm is generated. The counter loads TAUDnCDRm value into TAUDnCNTm and counts down.
- Slave channel:
  When INTTAUDnIm is generated on the master channel, the counter operation of the slave channel is triggered. The current value of TAUDnCDRm (slave) is loaded into TAUDnCNTm (slave) and the counter starts counting down from the TAUDnCDRm value. TAUDTTOUTm signal is set to the active level.
  When the counter reaches to $0000_H$ (duty time has elapsed), INTTAUDnIm is generated and a TAUDTTOUTm signal is set to an inactive level. The counter is reset to $FFFF_H$ and waits for the next INTTAUDnIm (start of the next pulse cycle) of the master channel.

The counter can stop operating by setting the TAUDnTT.TAUDnTTm of master and slave channels to 1. This sets TAUDnTE.TAUDnTEm to 0. TAUDnCNTm and TAUDTTOUTm of master and slave channels stop but their values are retained. The counter can be restarted by setting TAUDnTS.TAUDnTSm to 1.

**Conditions**

Simultaneous rewrite can be used with this function. See **Section 25.6, Simultaneous Rewrite**.

### 25.15.1.2 Equations

Pulse cycle = (TAUDnCDRm (master) + 1) × count clock cycle

Duty cycle [%] = (TAUDnCDRm (slave)/(TAUDnCDRm (master) + 1)) × 100

– Duty cycle = 0%
  TAUDnCDRm (slave) = $0000_H$
– Duty cycle = 100%
  TAUDnCDRm (slave) ≥ TAUDnCDRm (master) + 1

### 25.15.1.3 Block Diagram and General Timing Diagram

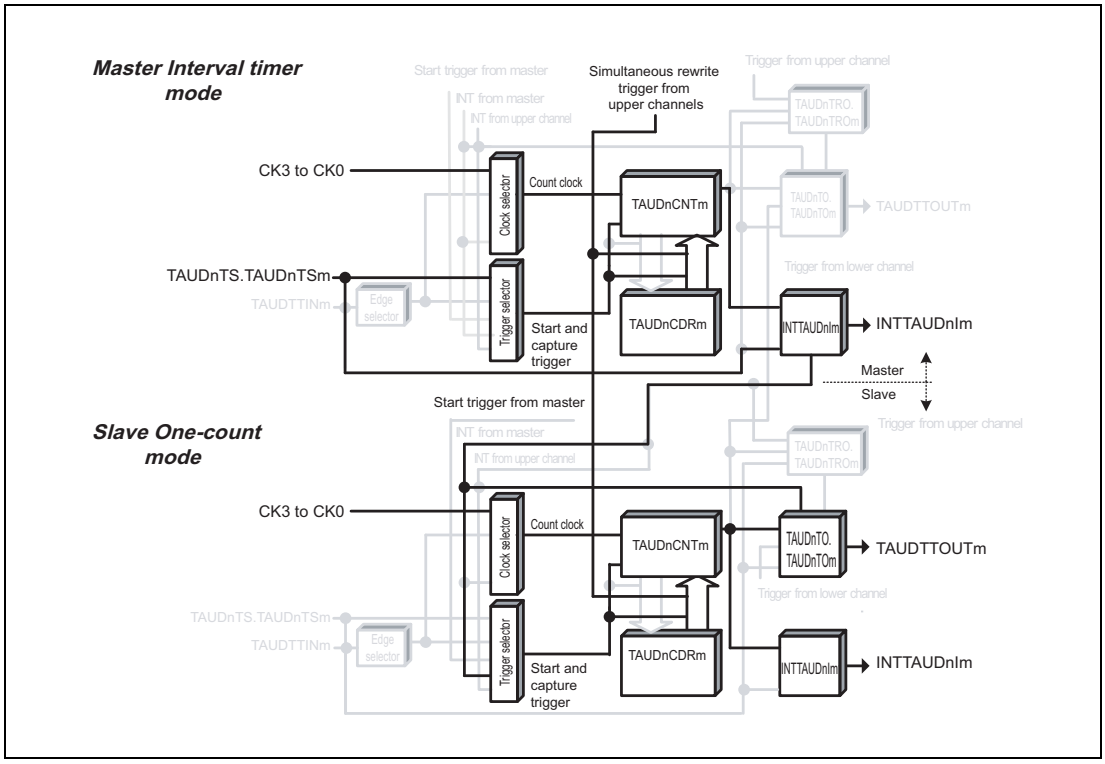



**Figure 25.90 Block Diagram of PWM Output Function**

The following settings apply to the general timing diagram.

- Slave channel: Positive logic (TAUDnTOL.TAUDnTOLm = 0)

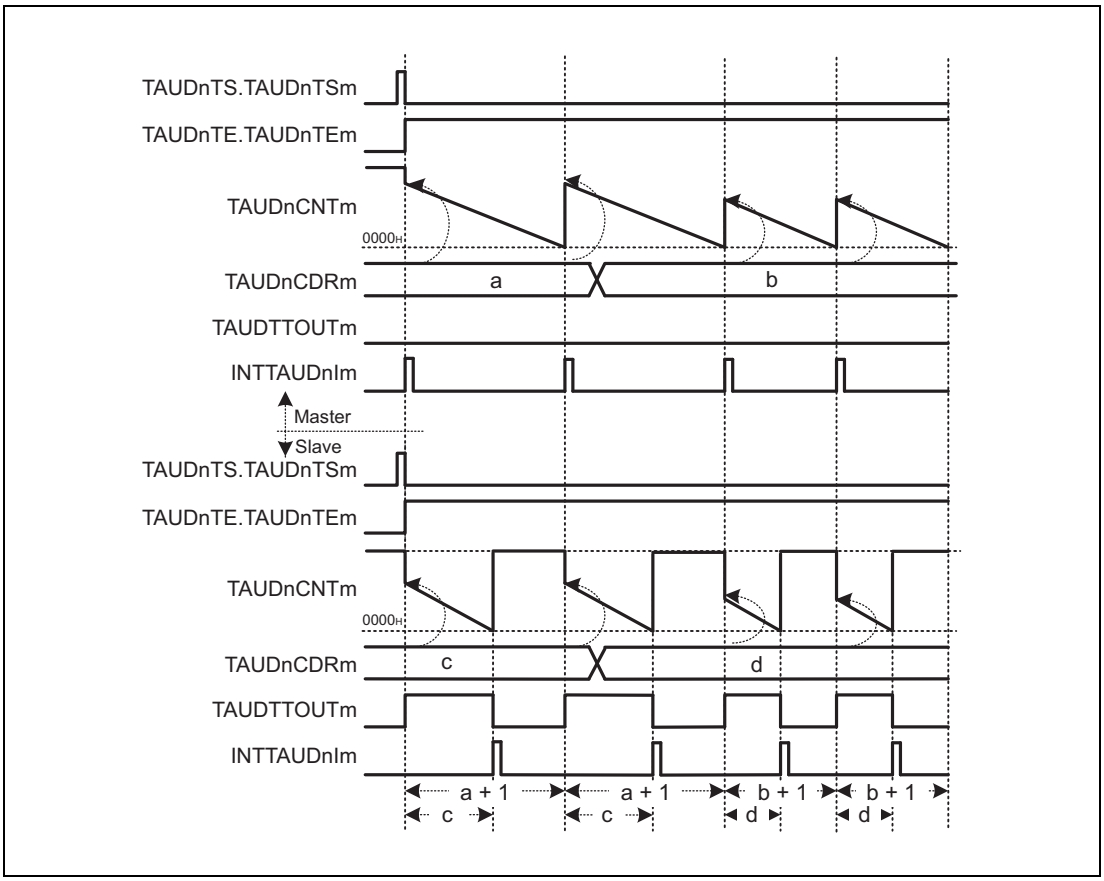



**Figure 25.91 General Timing Diagram of PWM Output Function**

**NOTES**

1. The interval between the starting to count and an interrupt being generated is the value of corresponding TAUDnCDRm + 1.
2. TAUDTTOUTm of the slave channel rises with a delay of one clock count after the rise of INTTAUDnIm of the master channel.

### 25.15.1.4 Register Settings for the Master Channel

#### (1) TAUDnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.142 Contents of the TAUDnCMORm Register for the Master Channel of the PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 1: Master channel |
| 10 to 8 | TAUDnSTS[2:0] | 000: Trigger the counter using software. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0000: Interval timer mode |
| 0 | TAUDnMD0 | 1: INTTAUDnIm generated at the beginning of operation. |

#### (2) TAUDnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.143 Contents of the TAUDnCMURm Register for the Master Channel of the PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for the master channel

The channel output mode is not used with this function.

### (4) Simultaneous rewrite for the master channel

Both the master and slave channels should have the same simultaneous rewrite settings.

**Table 25.144 Simultaneous Rewrite Settings for the Master Channel of the PWM Output Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Selects a master channel for simultaneous rewrite triggers.<br>1: Selects an upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 0: Generates a simultaneous rewrite trigger signal when the master channel starts to count. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

**NOTE**

Use with TAUDnRDS.TAUDnRDSm bit = 1 requires a channel higher than the master channel that operates with **Section 25.14.1, Simultaneous Rewrite Trigger Generation Function Type 1**.

Conduct operation settings under the following conditions:

- Simultaneous rewrite trigger output function type 1 setting channel: TAUDnRDCm = 1, TAUDnRDSm = 1
  TAUDnCDRm settings for this channel are as follows:
  = ((TAUDnCDR setting for the master channel targeted for simultaneous rewrite + 1) × interrupt count) – 1
- Master channel: TAUDnRDCm = 0, TAUDnRDSm = 1
- Slave channel: TAUDnRDCm = 0, TAUDnRDSm = 1

If TAUDnCDRm (slave) setting > TAUDnCDRm (master) setting + 1, the duty value (which exceeds 100%) is aggregated to be 100% output.

### 25.15.1.5 Register Settings for Slave Channels

#### (1) TAUDnCMORm for slave channels

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.145 Contents of the TAUDnCMORm Register for the Slave Channel of the PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 100: INTTAUDnIm of master channel is a start trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0100: One-count mode |
| 0 | TAUDnMD0 | 1: Start trigger during operation is valid. |

#### (2) TAUDnCMURm for slave channels

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.146 Contents of the TAUDnCMURm Register for the Slave Channel of the PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for slave channels

Table 25.147 Control Bit Settings in Synchronous Channel Output Mode 1

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 1: Synchronous channel operation |
| TAUDnTOC.TAUDnTOCm | 0: Operating mode 1 |
| TAUDnTOL.TAUDnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set these bits to 0 |
| TAUDnTRC.TAUDnTRCm | |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

### (4) Simultaneous rewrite for slave channels

Both the master and slave channels should have the same simultaneous rewrite settings.

Table 25.148 Simultaneous Rewrite Settings for Slave Channels of PWM Output Function

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Selects a master channel for simultaneous rewrite triggers.<br>1: Selects an upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 0: Generates a simultaneous rewrite trigger signal when the master channel starts to count. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

### 25.15.1.6 Operating Procedure for PWM Output Function

Table 25.149 Operating Procedure for PWM Output Function

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Master channel: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.1.4, Register Settings for the Master Channel**.<br><br>Slave channel: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.1.5, Register Settings for Slave Channels**.<br><br>Set the value of TAUDnCDRm register of every channel. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm of master and slave channels to 1 simultaneously.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm (master and slave channels) is set to 1 and the counters of master and slave channels start.<br>INTTAUDnIm is generated on the master channel and TAUDTTOUTm (slave) is set. |
| During operation | TAUDnCDRm can be changed at any time.<br>TAUDnTOL.TAUDnTOLm can be changed.<br>TAUDnCNTm and TAUDnRSF.TAUDnRSFm can be read at any time.<br><br>TAUDnRDT.TAUDnRDTm can be changed during operation. | TAUDnCNTm of master channel loads TAUDnCDRm value and counts down. When the counter reaches $0000_H$:<br>• INTTAUDnIm (master) is generated.<br>• TAUDnCDRm value is loaded into TAUDnCNTm (master) to continue count operation.<br>• TAUDnCDRm value is loaded into TAUDnCNTm (slave) to perform counting down.<br>• TAUDTTOUTm (slave) is set to the active level.<br>If TAUDnCNTm (slave) reaches $0000_H$:<br>• INTTAUDnIm (slave) is generated.<br>• TAUDTTOUTm (slave) is set to an inactive level. In addition, the counter of slave channel stops. |
| Stop Operation | Set TAUDnTT.TAUDnTTm of master and slave channels to 1 simultaneously.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm and TAUDTTOUTm stop and retain their current values. |

Restart Operation

### 25.15.1.7 Specific Timing Diagrams

#### (1) Duty cycle = 0%

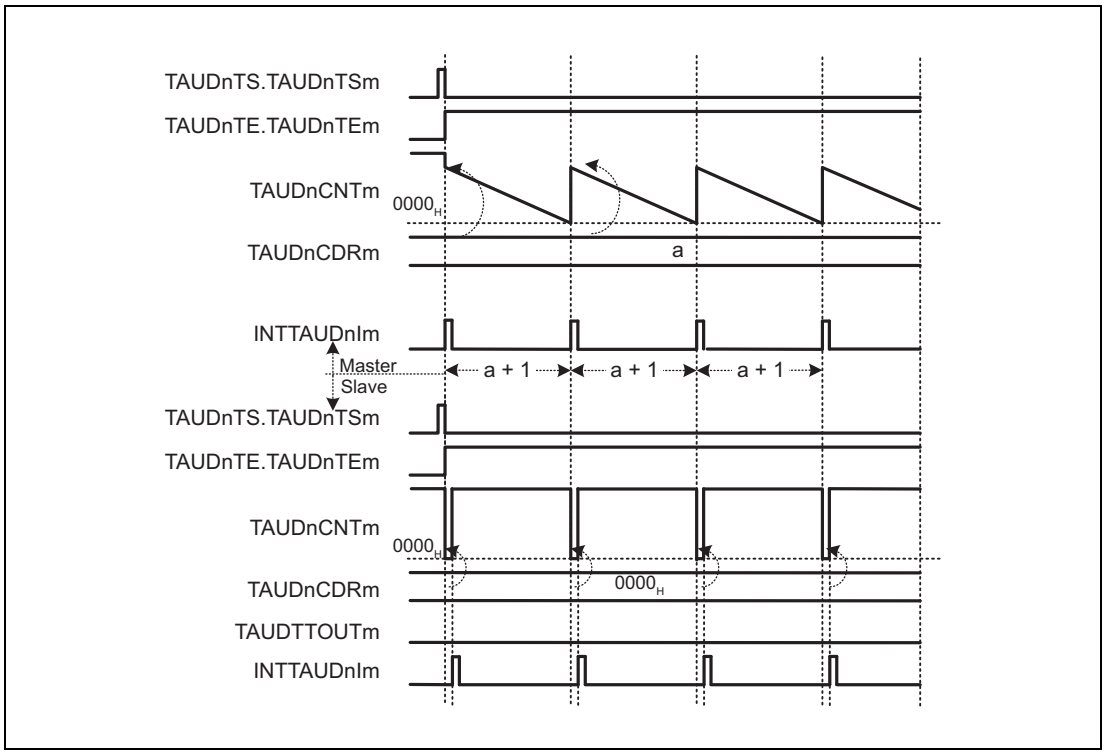



**Figure 25.92 TAUDnCDRm (Slave) = $0000_H$, Positive Logic (TAUDnTOL.TAUDnTOLm (Slave) = 0)**

- Every time the master channel generates an interrupt (INTTAUDnIm), $0000_H$ is loaded into TAUDnCNTm (slave). As a result, a slave channel interrupt (INTTAUDnIm) is generated at the same time and TAUDTTOUTm remains inactive.
- TAUDnCDRm value is loaded into TAUDnCNTm (slave) to generate an interrupt.

**(2) Duty cycle = 100%**

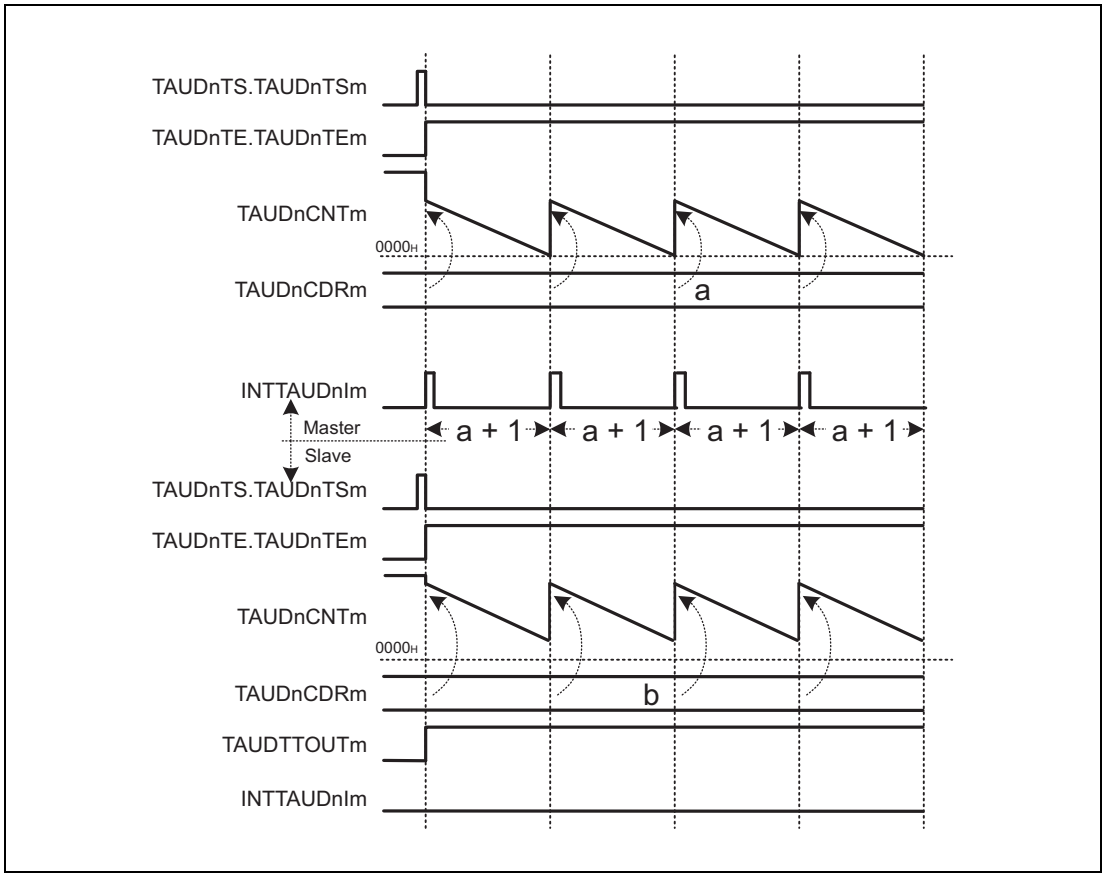



**Figure 25.93 TAUDnCDRm (Slave) ≥ TAUDnCDRm (Master) + 1 Positive Logic (TAUDnTOL.TAUDnTOLm (Slave) = 0)**

- If TAUDnCDRm (slave) value is greater than TAUDnCDRm (master) value, the slave channel counter does not reach $0000_H$ and consequently, no interrupt occurs. TAUDTTOUTm remains active.

### (3) Operation stop and restart

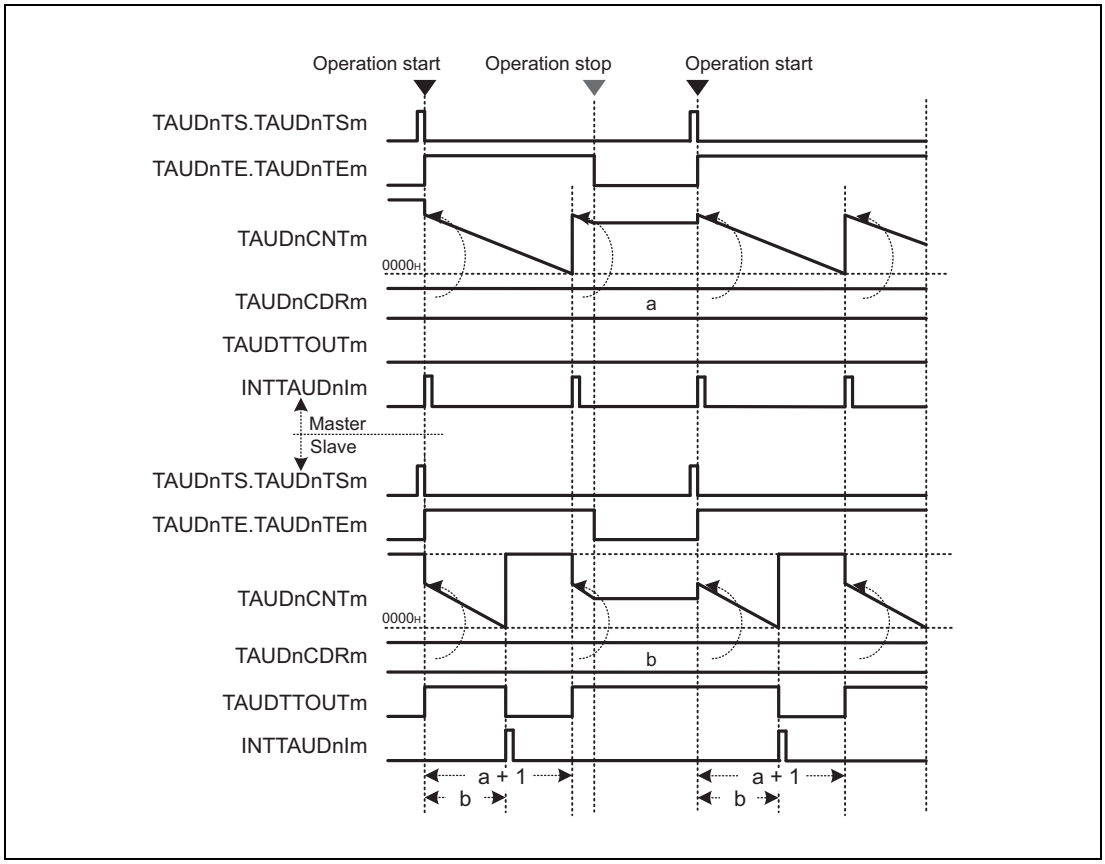



**Figure 25.94 Operation Stop and Restart**
**Positive Logic (TAUDnTOL.TAUDnTOLm (Slave) = 0)**

- The counter can be stopped by setting TAUDnTT.TAUDnTTm of master and slave channels to 1. This sets TAUDnTE.TAUDnTEm to 0.
- TAUDnCNTm and TAUDTTOUTm of all channels stop and the current values are retained. No interrupts are generated.
- The counter can be restarted by setting TAUDnTS.TAUDnTSm of master and slave channels to 1. TAUDnCDRm values of the master and slave channels are loaded to TAUDnCNTm and start to count down from these values.

## 25.15.2 One-Shot Pulse Output Function

### 25.15.2.1 Overview

#### Summary

This function outputs a signal pulse with a specific pulse width and delay time (both defined relative to an external input signal pulse) by using a master and a slave channel. The delay time is specified using the master channel. The pulse width is specified using the slave channel.

#### Prerequisites

- Two channels
- The operating mode for the master channel should be set to one-count mode. (See **Table 25.150, Contents of the TAUDnCMORm Register for the Master Channel of the One-Shot Pulse Output Function**.)
- The operating mode for slave channels should be set to pulse one-count mode. (See **Table 25.153, Contents of the TAUDnCMORm Register for the Slave Channel of the One-Shot Pulse Output Function**.)
- TAUDTTOUTm is not used with the master channel of this function.
- The channel output mode for the slave channel should be set to independent channel output mode 2. (See **Section 25.7, Channel Output Modes**.)
- TAUDTTINm (master) has to be detected while TAUDnCNTm (master) and TAUDnCNTm (slave) await a trigger. Furthermore, the slave is only triggered by an interrupt from the master channel and not by TAUDTTINm (slave).

#### Functional description

The counters are enabled by setting the channel trigger bits (TAUDnTS.TAUDnTSm) to 1 for the master and slave channels. This sets TAUDnTE.TAUDnTEm to 1, enabling count operation.

- Master channel:
  When the next valid TAUDTTINm input edge is detected, the current value of TAUDnCDRm is loaded into TAUDnCNTm. The counter starts to count down from this value. If TAUDnCMORm.TAUDnMD0 = 0, a trigger (TAUDTTINm) which is detected within the delay time is ignored.
  When the counter of master channel reaches $0000_H$, INTTAUDnIm is generated. The counter is reset to $FFFF_H$ and waits for the next valid TAUDTTINm input edge.
- Slave channel:
  INTTAUDnIm generated on master channel triggers the counter operation of slave channel. The current value of TAUDnCDRm (slave) is loaded into TAUDnCNTm (slave). The counter starts counting down from this value. An interrupt occurs and the TAUDTTOUTm signal is set.
  When the counter reaches $0001_H$, INTTAUDnIm is generated and TAUDTTOUTm signal is reset.
  The counter stops at $0000_H$ and waits for the next INTTAUDnIm of master channel.

The counter can be stopped by setting TAUDnTT.TAUDnTTm of master and slave channels to 1. This sets TAUDnTE.TAUDnTEm to 0. TAUDnCNTm and TAUDTTOUTm of master and slave channels stop but their values are retained. The counter can be restarted by setting TAUDnTS.TAUDnTSm to 1.

Setting TAUDnTS.TAUDnTSm to 1 while counting allows the counter to restart counting of master channel without making a stop (forced restart).

**Conditions**

- If TAUDnCMORn.TAUDnMD0 of master channel is set to 0, TAUDTTINm input edges detected during counting are ignored.
- Simultaneous rewrite can be used with this function. See **Section 25.6, Simultaneous Rewrite**.

### 25.15.2.2 Equations

Delay from trigger input to pulse output
= (TAUDnCDRm (master) + 1) × count clock cycle

Pulse width = (TAUDnCDRm (slave)) × count clock cycle

### 25.15.2.3 Block Diagram and General Timing Diagram

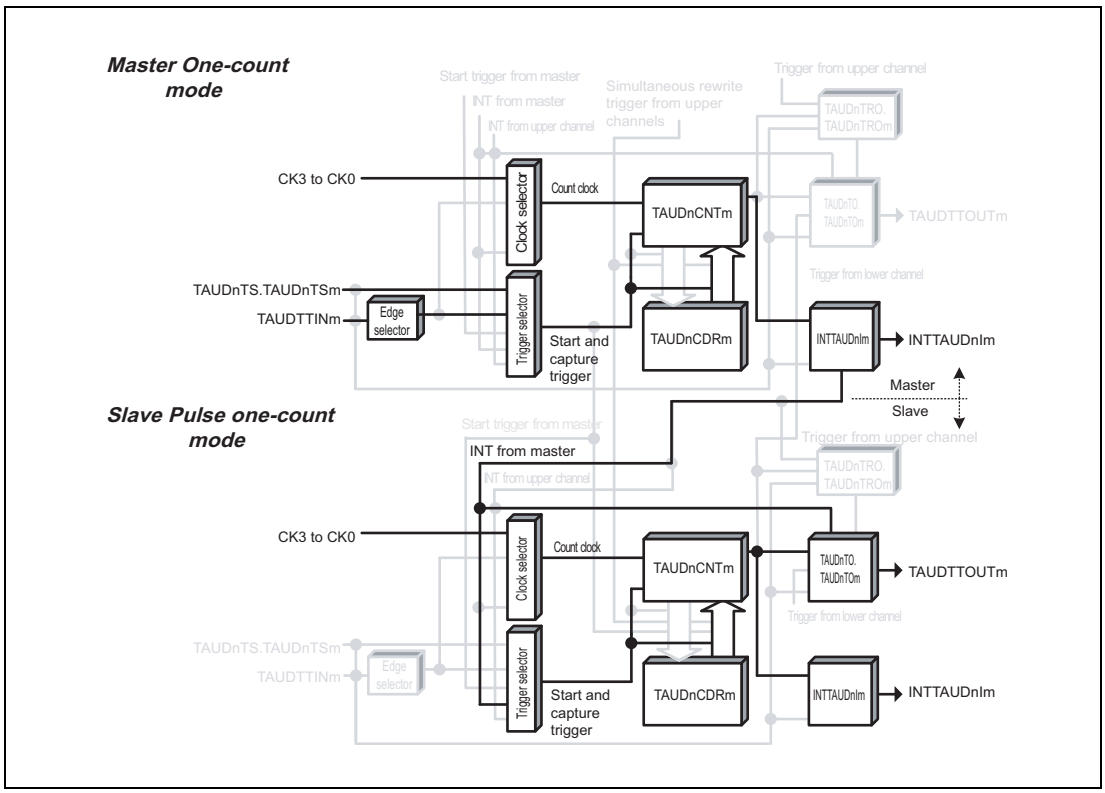



**Figure 25.95 Block Diagram of One-Shot Pulse Output Function**

The following settings apply to the general timing diagram.

- Start trigger detection is disabled during counting (TAUDnCMORm.TAUDnMD0 = 0).
- Detection of falling edge (TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)

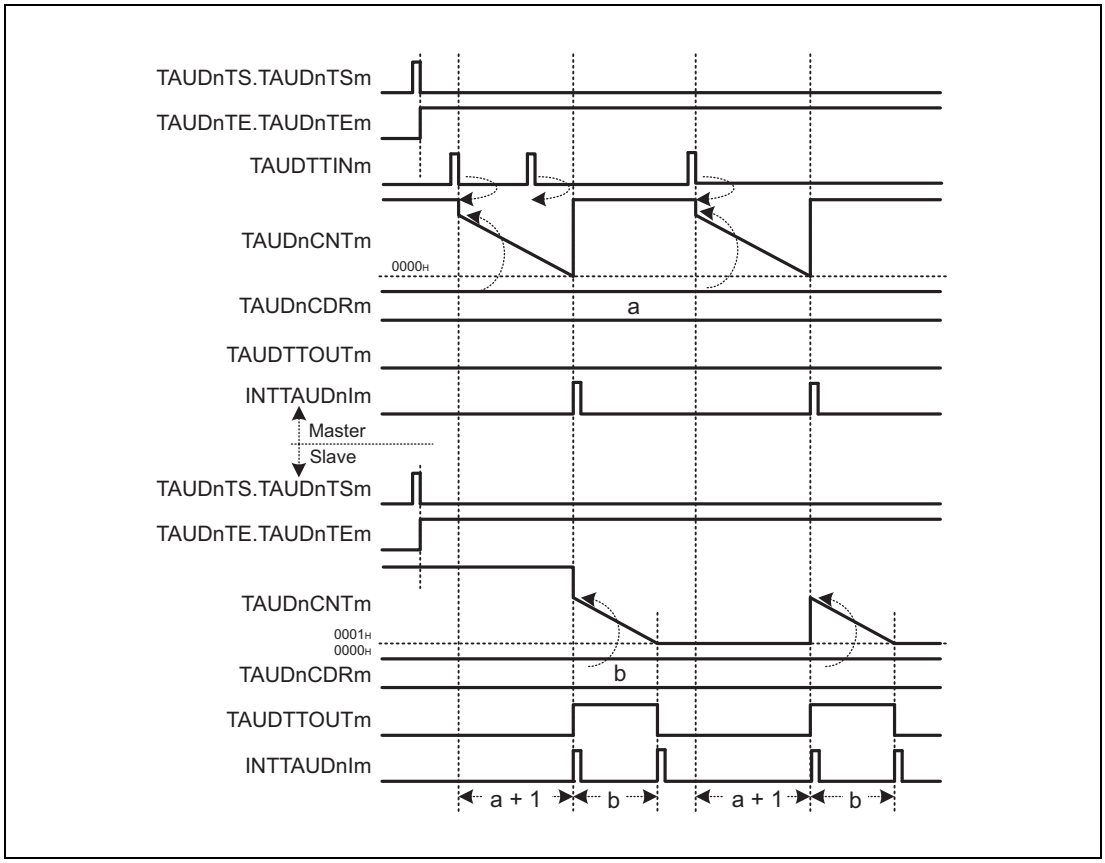



**Figure 25.96 General Timing Diagram of One-Shot Pulse Output Function**

### 25.15.2.4 Register Settings for the Master Channel

#### (1) TAUDnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.150 Contents of the TAUDnCMORm Register for the Master Channel of the One-Shot Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 1: Master channel |
| 10 to 8 | TAUDnSTS[2:0] | 001: Valid TAUDTTINm input edge signal is used as the start trigger |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0100: One-count mode |
| 0 | TAUDnMD0 | 0: Disables detection of start trigger during count operation.<br>1: Enables detection of start trigger during count operation.<br>The value of the MD0 bits of the master and slave channels must be identical. |

#### (2) TAUDnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.151 Contents of the TAUDnCMURm Register for the Master Channel of the One-Shot Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Detection of falling edge<br>01: Detection of rising edge<br>10: Detection of rising and falling edges<br>11: Setting prohibited |

### (3) Channel output mode for the master channel

TAUDnTOE.TAUDnTOEm is set to 0 because channel output mode is not used with this function.

### (4) Simultaneous rewrite for the master channel

Both the master and slave channels should have the same simultaneous rewrite settings.

**Table 25.152 Simultaneous Rewrite Settings for the Master Channel of One-Shot Pulse Output Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Master channel is simultaneous rewrite control channel. |
| TAUDnRDM.TAUDnRDMm | 0: Generates a simultaneous rewrite trigger signal when the master channel starts to count. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

### 25.15.2.5 Register Settings for Slave Channels

#### (1) TAUDnCMORm for slave channels

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.153 Contents of the TAUDnCMORm Register for the Slave Channel of the One-Shot Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 100: INTTAUDnIm of master channel is a start trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 1010: Pulse one-count mode |
| 0 | TAUDnMD0 | 0: Disables detection of start trigger during count operation.<br>1: Enables start trigger detection while counting.<br>The MD0 bits of the master and slave channels must be indentical. |

#### (2) TAUDnCMURm for slave channels

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.154 Contents of the TAUDnCMURm Register for the Slave Channel of the One-Shot Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for the slave channel

Table 25.155 Control Bit Settings in Independent Channel Output Mode 2

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 0: Independent channel output |
| TAUDnTOC.TAUDnTOCm | 1: Operating mode 2 |
| TAUDnTOL.TAUDnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set this bit to 0 |
| TAUDnTDL.TAUDnTDLm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set this bit to 0 |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set this bit to 0 |
| TAUDnTRC.TAUDnTRCm | 0: Disables the operation as a real-time output trigger channel |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

### (4) Simultaneous rewrite for slave channels

Both the master and slave channels should have the same simultaneous rewrite settings.

Table 25.156 Simultaneous Rewrite Settings for Slave Channels of One-Shot Pulse Output Function

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Master channel is simultaneous rewrite control channel. |
| TAUDnRDM.TAUDnRDMm | 0: Generates a simultaneous rewrite trigger signal when the master channel starts to count. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

### 25.15.2.6 Operating Procedure for One-Shot Pulse Output Function

Table 25.157 Operating Procedure for One-Shot Pulse Output Function

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Master channel: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.2.4, Register Settings for the Master Channel**.<br><br>Slave channel: Set TAUDnCMORm and TAUDnCMURm registers and channel output mode as described in **Section 25.15.2.5, Register Settings for Slave Channels**.<br><br>Set the value of TAUDnCDRm register of every channel. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm of master and slave channels to 1 simultaneously.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm (master and slave channels) is set to 1 and the master channel awaits a TAUDTTINm input. |
| During Operation | TAUDnCDRm can be changed at any time. TAUDnCNTm and TAUDnRSF.TAUDnRSFm can be read at any time.<br><br>TAUDnRDT.TAUDnRDTm can be changed during operation. | When valid TAUDTTINm input edge is detected, TAUDnCDRm value of master channel is loaded into TAUDnCNTm to perform counting down. When the counter reaches $0000_H$:<br>• INTTAUDnIm (master) is generated.<br>• TAUDnCNTm (master) is reset to $FFFF_H$ and waits for the next valid TAUDTTINm input edge.<br>• TAUDnCDRm value is reloaded into TAUDnCNTm (slave) to perform counting down.<br>• INTTAUDnIm (slave) is generated.<br>• TAUDTTOUTm (slave) is set to the active level.<br>When TAUDnCNTm (slave) reaches $0001_H$:<br>• INTTAUDnIm (slave) is generated.<br>• TAUDTTOUTm (slave) is set to an inactive level. In addition, the counter of slave channel stops. |
| Stop Operation | Set TAUDnTT.TAUDnTTm of master and slave channels to 1 simultaneously.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm and TAUDTTOUTm stop and retain their current values. |

Restart Operation

### 25.15.2.7 Specific Timing Diagrams

#### (1) TAUDnCDRm (master) = $0000_H$

The following settings apply to this diagram.

- Disables detection of start trigger during count operation. (TAUDnCMORm.TAUDnMD0 = 0)
- Detection of falling edge (TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)

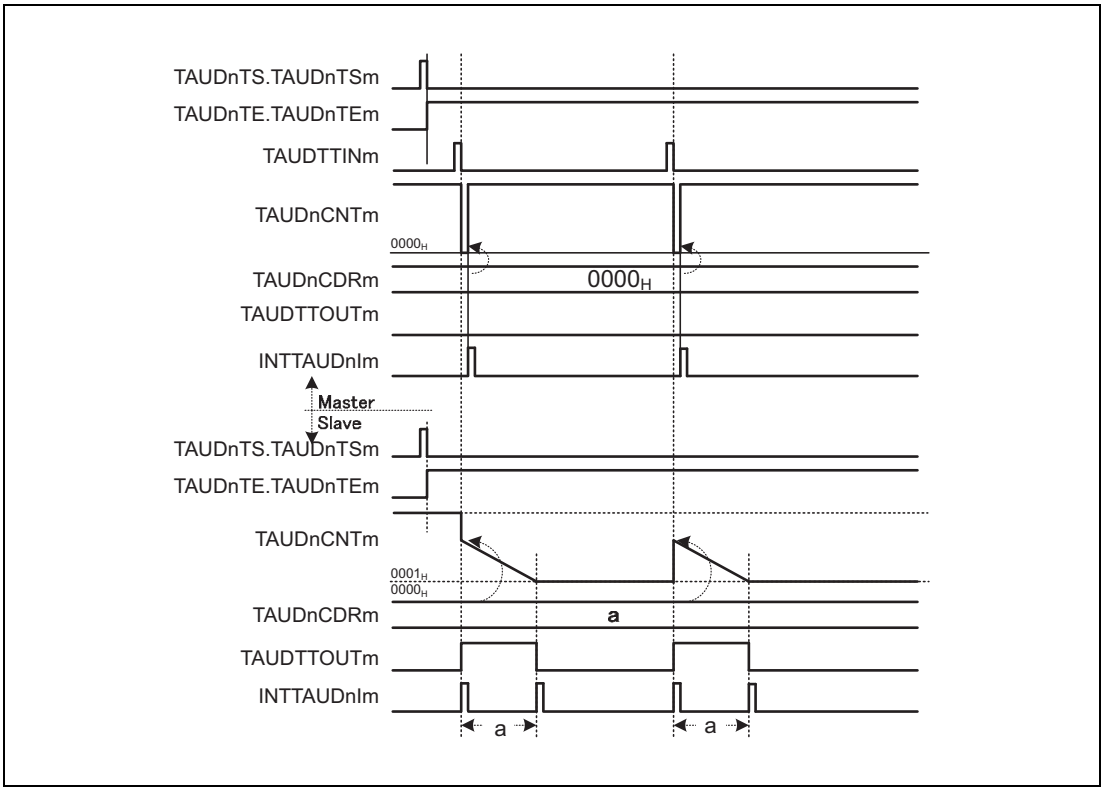



**Figure 25.97 TAUDnCDRm (Master) = $0000_H$**

- When a valid TAUDTTINm input edge is detected, the value $0000_H$ is written to TAUDnCNTm (master). The counter is set to $0000_H$ for one count and returns to $FFFF_H$.
  Thus the slave channel starts to count down one count clock later than TAUDTTINm (master).

### (2) TAUDnCDRm (slave) = $0000_H$

The following settings apply to this diagram.

- Disables detection of start trigger during count operation. (TAUDnCMORm.TAUDnMD0 = 0)
- Detection of falling edge (TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)

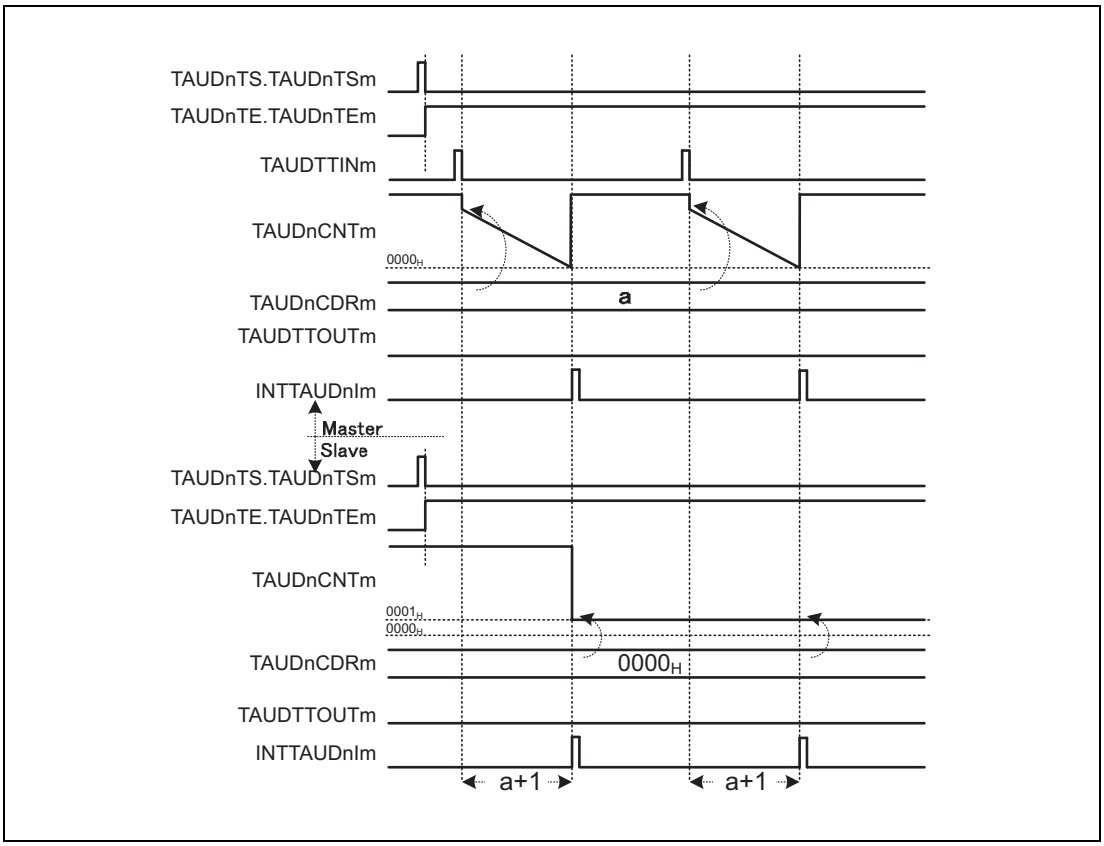



**Figure 25.98 TAUDnCDRm (Slave) = $0000_H$**

- TAUDTTOUTm remains inactive, because the pulse width is zero.

### (3) TAUDnCMORm.TAUDnMD0 = 1

The following settings apply to this diagram.

- Enables start trigger detection while counting. (TAUDnCMORm.TAUDnMD0 = 1)
- Detection of falling edge (TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)

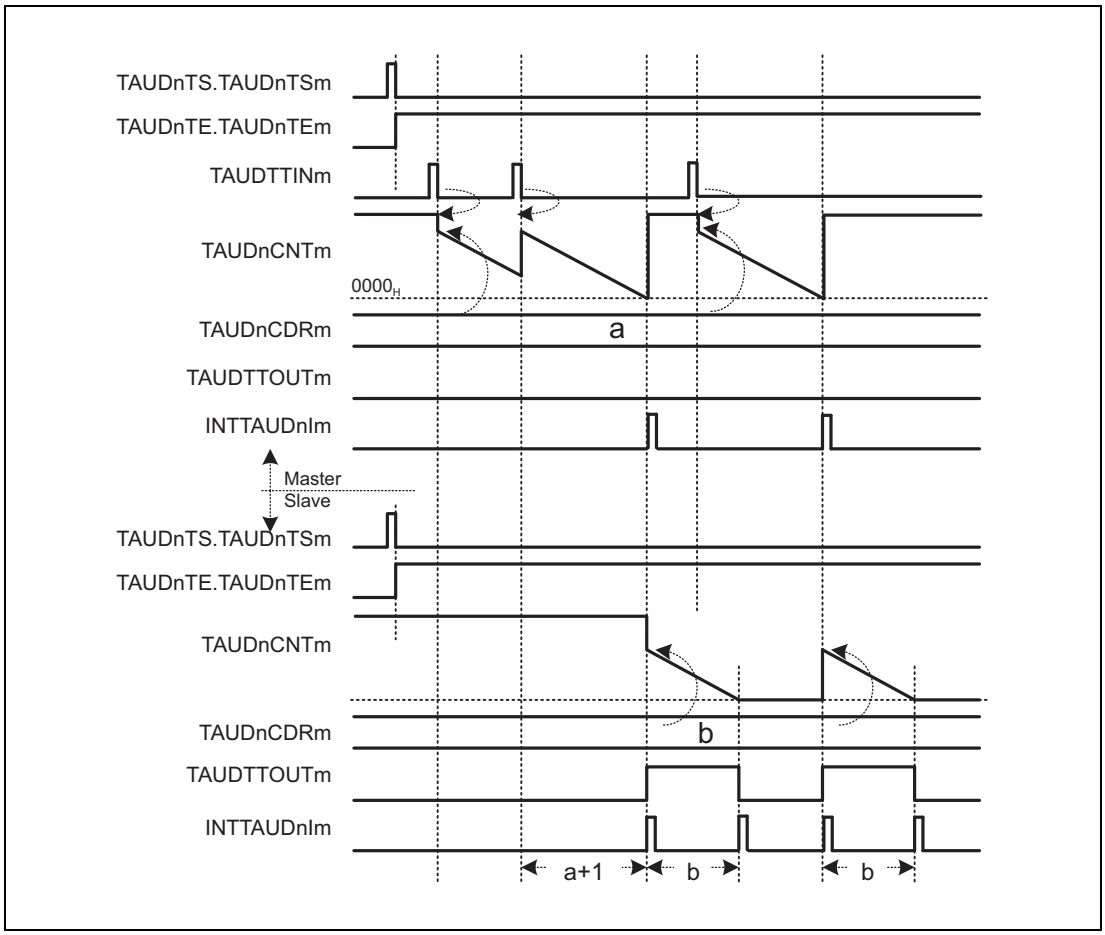



**Figure 25.99 TAUDnCMORm.TAUDnMD0 = 1**

- If a valid TAUDTTINm input edge is detected while the counter of the master channel counts down, TAUDnCNTm reloads the value of TAUDnCDRm. The counter restarts to count down. This means the delay is extended by the value of TAUDnCNTm at the time when a valid TAUDTTINm input edge is detected.

### (4) Restarting the master channel while the slave channel is counting

The following settings apply to this diagram.

- Disables detection of start trigger during count operation. (TAUDnCMORm.TAUDnMD0 = 0)
- Detection of falling edge (TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)

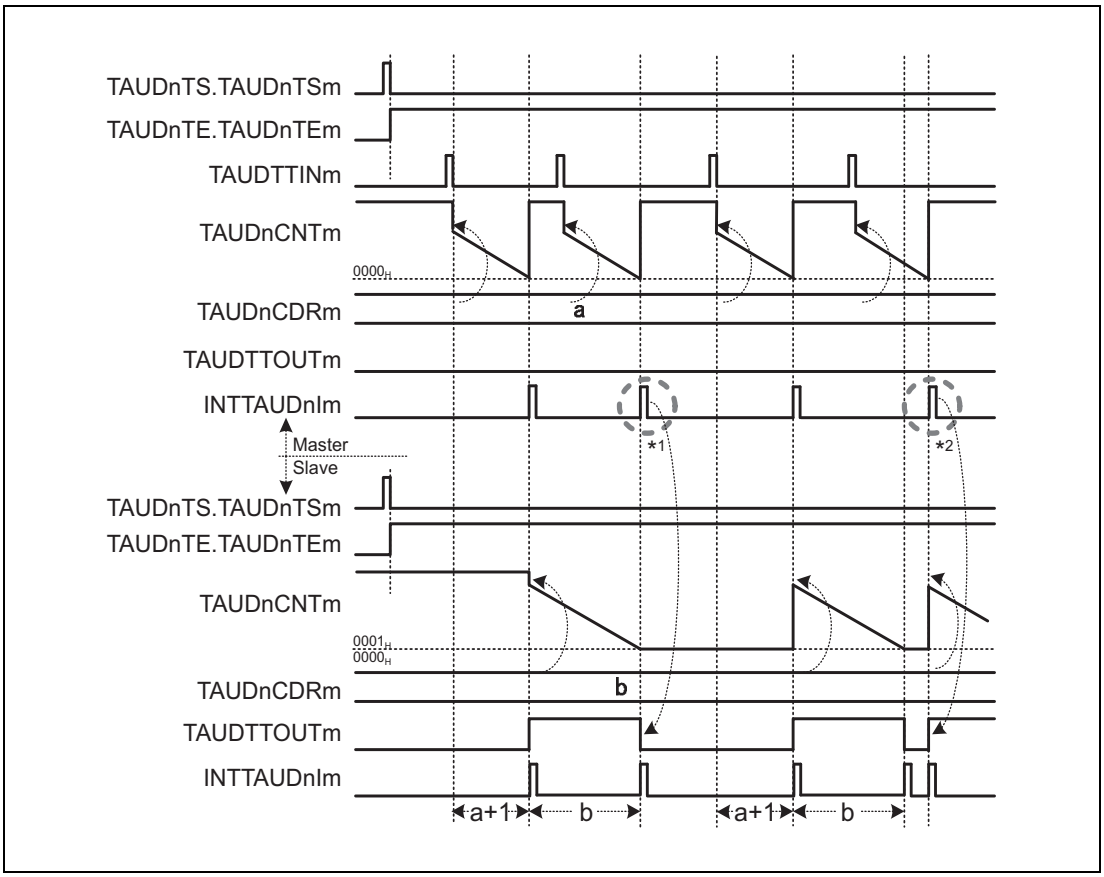



**Figure 25.100 TAUDTTINm input interval ≤ Delay Time + Pulse Width +1**

- If the master channel generates an interrupt before the counter of the slave channel has reached $0001_H$ or exactly when $0001_H$ is reached[*1], the interrupt (master) is ignored.
- If an interrupt of the master channel occurs when the counter of the slave channel awaits the next trigger, the value of TAUDnCDRm (slave) is reloaded. An interrupt is generated and TAUDTTOUTm toggles. If TAUDnCNTm (master) has started to count down while the TAUDnCNTm (slave) is still counting[*2], TAUDTTOUTm is not output with the expected delay time.
- To generate the correct one-shot pulse, the start trigger for the master channel must be detected while the master and slave channels are waiting for the start trigger, and not while they are counting.

## 25.15.3 Trigger Start PWM Output Function

### 25.15.3.1 Overview

#### Summary

This function generates a PWM output using a master and a slave channel. It enables the pulse cycle (frequency) and the duty of the TAUDTTOUTm to be set. The pulse cycle is specified using the master channel. The duty is specified using the slave channel. The Trigger Start PWM Output Function is identical to PWM Output Function except that the master channel of this function can be reset by a valid TAUDTTINm input edge.

#### Prerequisites

- Two channels
- The operation mode of the master channel must be set to Interval Timer Mode (see **Table 25.158, Contents of the TAUDnCMORm Register for the Master Channel of the Trigger Start PWM Output Function**).
- The operation mode of the slave channel must be set to One-Count Mode (see **Table 25.161, Contents of the TAUDnCMORm Register for the Slave Channel of the Trigger Start PWM Output Function**).
- The channel output mode of the slave channel must be set to Synchronous Channel Output Mode 1 (see **Section 25.7, Channel Output Modes**).
- TAUDTTOUTm is not used with the master channel of this function.

#### Functional description

The counters (master and slave) are enabled by setting the channel trigger bits (TAUDnTS.TAUDnTSm) to 1. This in turn sets TAUDnTE.TAUDnTEm to 1, enabling count operation. The current value of TAUDnCDRm is loaded to TAUDnCNTm, and the counter starts to count down from this value. INTTAUDnIm is generated on the master channel, and a PWM output is realized by setting and resetting TAUDTTOUTm (slave).

- Master channel:
  The current value of TAUDnCDRm is loaded to the counter (TAUDnCNTm), INTTAUDnIm is generated and the counter starts to count down from this value.
  When the counter reaches $0000_H$ and the pulse cycle time has elapsed, INTTAUDnIm is generated and the counters (master and slave) load the current TAUDnCDRm values.
  If a valid TAUDTTINm input edge is detected, the counter of the master channel loads the current TAUDnCDRm value, restarts counting down and generates an interrupt.
- Slave channel:
  When the slave detects an interrupt from the master channel, it starts to count down from the current value of TAUDnCDRm. The TAUDTTOUTm signal is set to the active level.
  When the counter reaches $0000_H$ (duty time has elapsed), INTTAUDnIm is generated and the TAUDTTOUTm signal is reset. The counter returns to $FFFF_H$ and awaits the next INTTAUDnIm of the master channel.

The counter can be stopped by setting TAUDnTT.TAUDnTTm to 1 for the master and slave channels, which in turn sets TAUDnTE.TAUDnTEm to 0. TAUDnCNTm and TAUDTTOUTm of master and slave channels stop but retain their values. The counters can be restarted by setting TAUDnTS.TAUDnTSm to 1.

**Conditions**

Simultaneous rewrite can be used with this function. See **Section 25.6, Simultaneous Rewrite**.

### 25.15.3.2 Equations

Pulse cycle = (TAUDnCDRm (master) + 1) × count clock cycle

Duty cycle [%] = [TAUDnCDRm (slave) / (TAUDnCDRm (master) + 1)] × 100

– Duty cycle = 0%
  TAUDnCDRm (slave) = $0000_H$
– Duty cycle = 100%
  TAUDnCDRm (slave) ≥ TAUDnCDRm (master) + 1

### 25.15.3.3 Block Diagram and General Timing Diagram

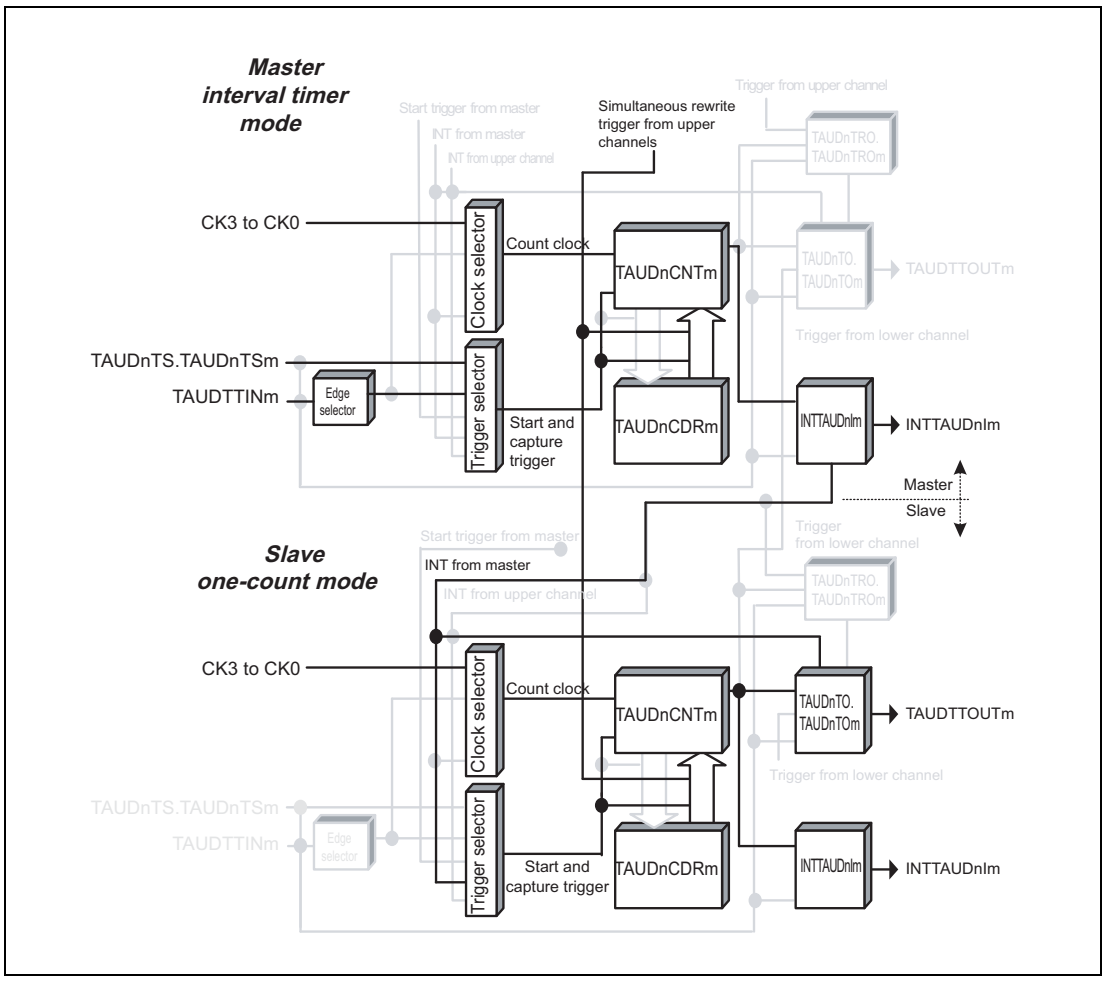



**Figure 25.101 Block Diagram for Trigger Start PWM Output Function**

The following settings apply to the general timing diagram.

- Detection of rising edge (TAUDnCMURm.TAUDnTIS[1:0] = $01_B$)
- Positive logic (TAUDnTOL.TAUDnTOLm (slave) = 0)

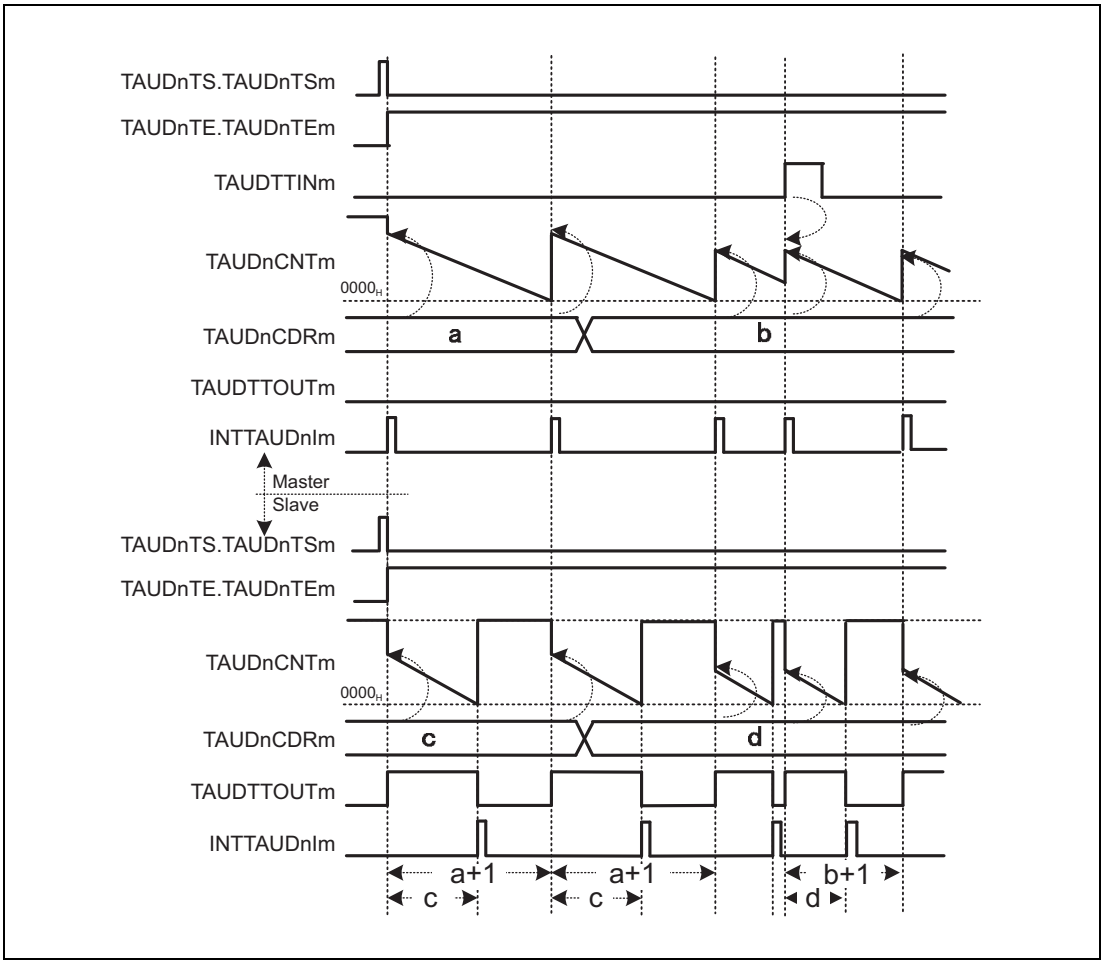



**Figure 25.102 General Timing Diagram for Trigger Start PWM Output Function**

**NOTE**

TAUDTTOUTm of the slave channel rises with a delay of one clock count after the rise of INTTAUDnIm of the master channel.

### 25.15.3.4 Register Settings for the Master Channel

#### (1) TAUDnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.158 Contents of the TAUDnCMORm Register for the Master Channel of the Trigger Start PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 1: Master channel |
| 10 to 8 | TAUDnSTS[2:0] | 001: Valid TAUDTTINm input edge signal is used as the start trigger |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0000: Interval timer mode |
| 0 | TAUDnMD0 | 1: INTTAUDnIm generated at the beginning of operation. |

#### (2) TAUDnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.159 Contents of the TAUDnCMURm Register for the Master Channel of the Trigger Start PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Detection of falling edge<br>01: Detection of rising edge<br>10: Detection of rising and falling edges<br>11: Setting prohibited |

### (3) Channel output mode for the master channel

The channel output mode is not used by this function.

### (4) Simultaneous rewrite for the master channel

Both the master and slave channels should have the same simultaneous rewrite settings.

**Table 25.160 Simultaneous Rewrite Settings for the Master Channel of the Trigger Start PWM Output Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Selects a master channel for simultaneous rewrite triggers.<br>1: Selects an upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 0: Generates a simultaneous rewrite trigger signal when the master channel starts to count. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

### 25.15.3.5 Register Settings for Slave Channels

#### (1) TAUDnCMORm for slave channels

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.161 Contents of the TAUDnCMORm Register for the Slave Channel of the Trigger Start PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 100: INTTAUDnIm of master channel is a start trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0100: One-count mode |
| 0 | TAUDnMD0 | 1: Start trigger during operation is valid.<br>The value of the TAUDnMD[0] bit of the master and slave channels must be identical. |

#### (2) TAUDnCMURm for slave channels

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.162 Contents of the TAUDnCMURm Register for the Slave Channel of the Trigger Start PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for the slave channel

Table 25.163 Control Bit Settings in Synchronous Channel Output Mode 1

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 1: Synchronous channel operation |
| TAUDnTOC.TAUDnTOCm | 0: Operating mode 1 |
| TAUDnTOL.TAUDnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set this bit to 0 |
| TAUDnTRC.TAUDnTRCm | 0: Disables the operation as a real-time output trigger channel |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

### (4) Simultaneous rewrite for slave channels

Both the master and slave channels should have the same simultaneous rewrite settings.

Table 25.164 Simultaneous Rewrite Settings for the Slave Channel of the Trigger Start PWM Output Function

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Selects a master channel for simultaneous rewrite triggers.<br>1: Selects an upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 0: Generates a simultaneous rewrite trigger signal when the master channel starts to count. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

### 25.15.3.6 Operating Procedure for Trigger Start PWM Output Function

Table 25.165 Operating Procedure for Trigger Start PWM Output Function

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Master channel: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.3.4, Register Settings for the Master Channel**.<br><br>Slave channel: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.3.5, Register Settings for Slave Channels**.<br><br>Set the value of TAUDnCDRm register of every channel. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm of master and slave channels to 1 simultaneously.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm (master and slave channels) is set to 1 and the counters of master and slave channels start.<br>INTTAUDnIm is generated on the master channel. |
| During Operation | TAUDnCDRm can be changed at any time. TAUDnCNTm and TAUDnRSF.TAUDnRSFm can be read at any time.<br><br>TAUDnRDT.TAUDnRDTm can be changed during operation. | TAUDnCNTm of master channel loads TAUDnCDRm value and counts down. When the counter reaches $0000_H$:<br>• INTTAUDnIm (master) is generated.<br>• TAUDnCDRm value is loaded into TAUDnCNTm (master) to continue count operation.<br>• TAUDnCNTm (slave) loads the TAUDnCDRm value and starts to count down<br>• TAUDTTOUTm (slave) is set<br>When TAUDnCNTm of the slave = $0000_H$:<br>• INTTAUDnIm (slave) is generated.<br>• TAUDTTOUTm (slave) is set to an inactive level. In addition, the counter of slave channel stops.<br><br>If a TAUDTTINm input is detected on the master channel while TAUDnCNTm of the master channel is counting down:<br>• TAUDnCNTm (master and slave) loads the TAUDnCDRm value and counts down<br>• INTTAUDnIm (master) is generated.<br>• TAUDTTOUTm (slave) is set to the active level. |
| Stop Operation | Set TAUDnTT.TAUDnTTm of master and slave channels to 1 simultaneously.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm and TAUDTTOUTm stop and retain their current values. |

Restart Operation

### 25.15.3.7 Specific Timing Diagrams

#### (1) Duty cycle = 0%

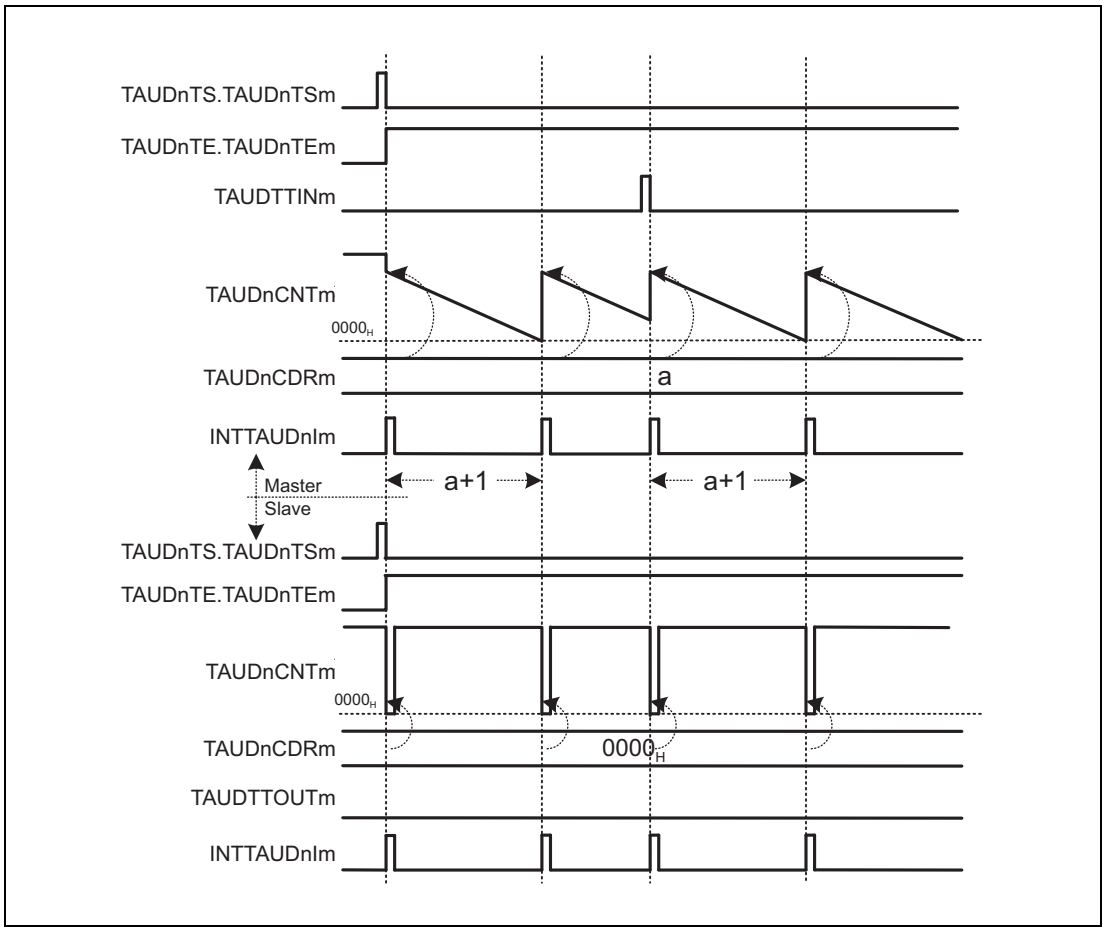



**Figure 25.103 TAUDnCDRm (Slave) = $0000_H$, Positive Logic (TAUDnTOL.TAUDnTOLm (Slave) = 0) Detection of Falling Edge (TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)**

- Every time the master channel generates an interrupt (INTTAUDnIm), $0000_H$ is written to TAUDnCNTm (slave). Therefore, TAUDnCNTm (slave) cannot start to count and TAUDTTOUTm remains inactive.
- TAUDnCNTm (slave) generates an interrupt every time the value of TAUDnCDRm is reloaded. The detection of a valid TAUDTTINm input edge has no effect on TAUDTTOUTm (slave).

**(2) Duty cycle = 100%**

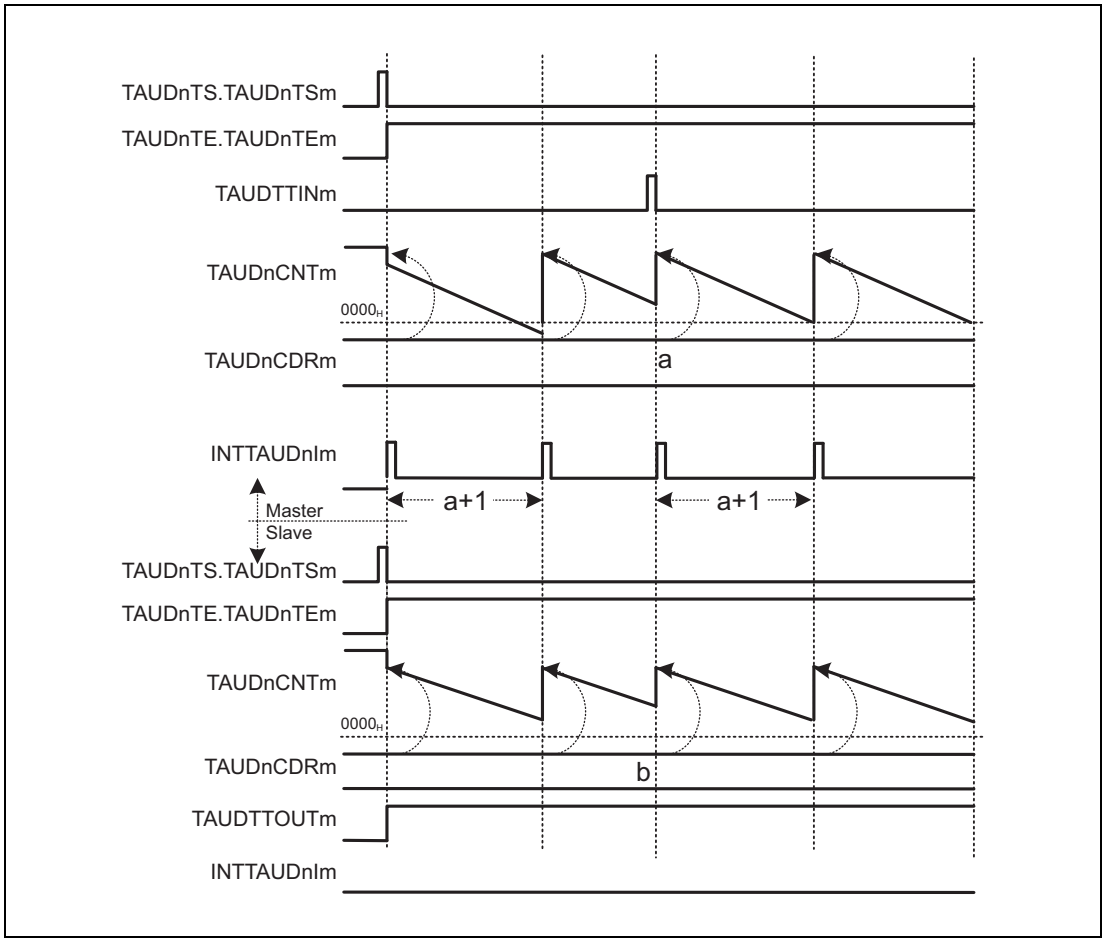



**Figure 25.104 TAUDnCDRm (Slave) ≥ TAUDnCDRm (Master) + 1, Positive Logic (TAUDnTOL.TAUDnTOLm (Slave) = 0) Falling Edge Detection (TAUDnCMURm.TIS[1:0] = $00_B$)**

- If the value TAUDnCDRm (slave) is higher than the value TAUDnCDRm (master), the counter of the slave channel cannot reach $0000_H$ and cannot generate interrupts.
  The TAUDTTOUTm remains at active state.
  The detection of a valid TAUDTTINm input edge has no effect on TAUDTTOUTm (slave).

### (3) TAUDTTINm detection and active slave counter

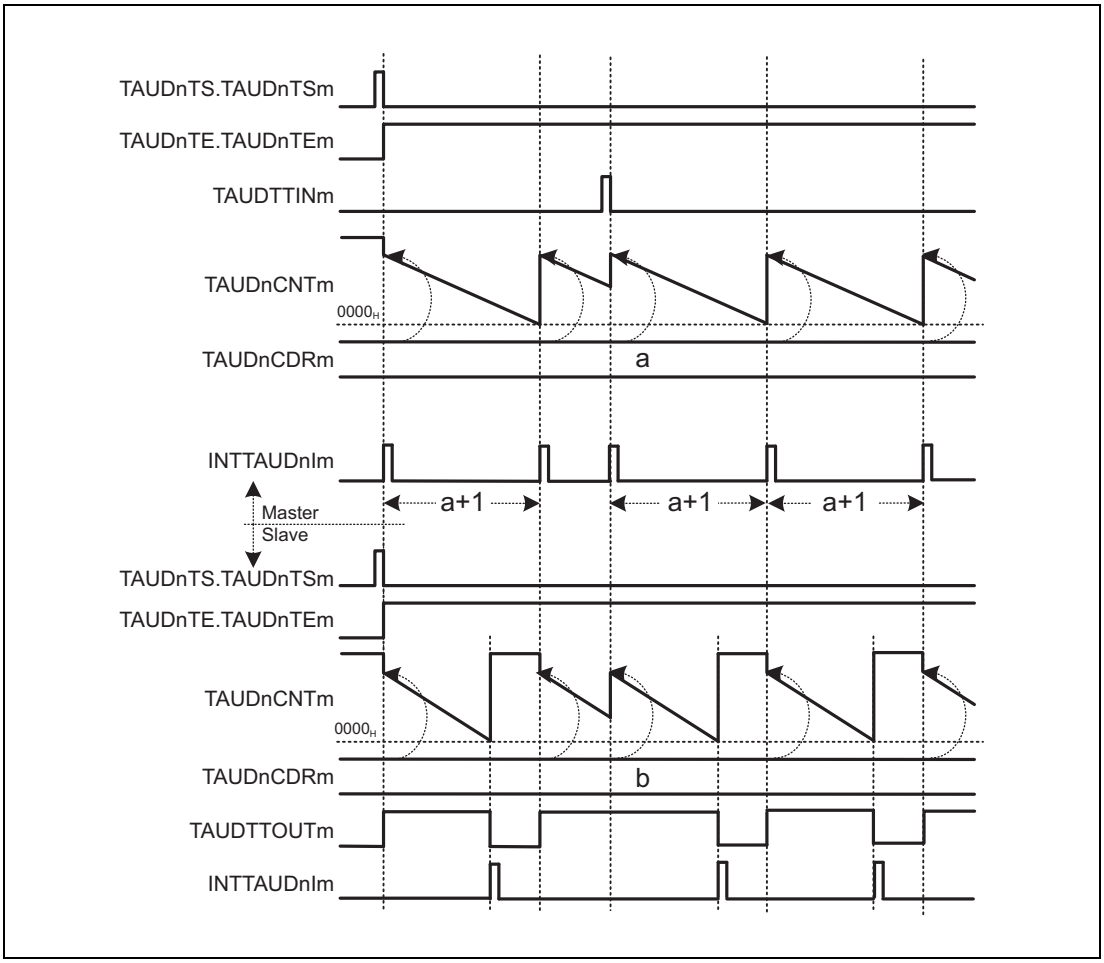



**Figure 25.105 Positive Logic (TAUDnTOL.TAUDnTOLm (Slave) = 0)**
**Detection of Falling Edge (TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)**

- If TAUDnCNTm (slave) reloads the value TAUDnCDRm (slave) while it is still counting down, TAUDTTOUTm remains unchanged and extends the duty.
  The duty does not correspond to the value of the slave's data register.

## 25.15.4 Delay Pulse Output Function

### 25.15.4.1 Overview

#### Summary

This function outputs two signals. The pulse width and pulse cycle of the reference signal are defined using the master channel and slave channel 1. Slave channels 2 and 3 output the reference signal with a specified delay. The delay signal is identical to the reference signal, but delayed by the amount specified on slave channel 2.

The signal values are specified in the following way:

- The pulse cycle is specified using the master channel.
- The duty cycle of the reference signal is specified using slave channel 1. The duty cycle of the delay signal is specified using slave channel 3.
- The delay is specified on slave channel 2.

#### Prerequisites

- Four channels
- The operating mode for the master channel should be set to interval timer mode. (See **Table 25.166, Contents of the TAUDnCMORm Register for the Master Channel of the Delay Pulse Output Function**.)
- The operating mode for slave channels 1 and 2 should be set to one-count mode. (See **Table 25.169, Contents of the TAUDnCMORm Register for Slave Channel 1 of the Delay Pulse Output Function** and **Table 25.173, Contents of the TAUDnCMORm Register for Slave Channel 2 of the Delay Pulse Output Function**.)
- The operating mode for slave channel 3 should be set to pulse one-count mode. (See **Table 25.176, Contents of the TAUDnCMORm Register for Slave Channel 3 of the Delay Pulse Output Function**.)
- TAUDTTOUTm is not used with the master channel and slave channel 2.
- The channel output mode for slave channel 1 should be set to synchronous channel output mode 1. (See **Section 25.7, Channel Output Modes**.)
- The channel output mode for slave channel 3 should be set to independent channel output mode 2. (See **Section 25.7, Channel Output Modes**.)

#### Functional description

The counters of the channel group are enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This sets TAUDnTE.TAUDnTEm to 1, enabling count operation.

- Master channel:
  The current value of TAUDnCDRm is loaded into TAUDnCNTm and the counter starts to count down from this value. INTTAUDnIm is generated on the master channel.
  When the counter value of master channel reaches $0000_H$ and pulse cycle time has elapsed, INTTAUDnIm is generated. The TAUDnCDRm value is reloaded into the counter to perform counting down.
- Slave channels 1 and 2:
  Slave channels 1 and 2 start to count down from the current TAUDnCDRm value when detecting

an interrupt from the master channel. TAUDTTOUTm signal (slave 1) is set.

– Slave channel 1:
When the counter of slave channel 1 reaches $0000_H$ (duty time has elapsed), INTTAUDnIm is generated and TAUDTTOUTm signal is reset. The counter is reset to $FFFF_H$ and waits for the next INTTAUDnIm of master channel.

– Slave channel 2:
When the counter of slave channel 2 reaches $0000_H$ and delay time has elapsed, INTTAUDnIm is generated. The counter is reset to $FFFF_H$ and waits for the next INTTAUDnIm of master channel.
Generating INTTAUDnIm (slave channel 2) triggers the counter of slave channel 3.

- Slave channel 3:
When slave channel 3 detects an interrupt from slave channel 2, its counter starts counting down from the current value of TAUDnCDRm. INTTAUDnIm is generated and the TAUDTTOUTm signal (slave channel 3) is set.
When the counter of slave channel 3 reaches $0001_H$, INTTAUDnIm is generated and the TAUDTTOUTm signal is reset.
The delayed PWM pulse is output from slave channel 3.

The counter can be stopped by setting TAUDnTT.TAUDnTTm of master and slave channels to 1. This sets TAUDnTE.TAUDnTEm to 0. TAUDnCNTm and TAUDTTOUTm of master and slave channels stop but their values are retained. The counter can be restarted by setting TAUDnTS.TAUDnTSm to 1.

**Conditions**

Simultaneous rewrite can be used with this function. See **Section 25.6, Simultaneous Rewrite**.

### 25.15.4.2 Equations

Pulse cycle = (TAUDnCDRm (master) + 1) × count clock cycle

Duty width 1 = (TAUDnCDRm (slave 1)) × count clock cycle

Delay width = (TAUDnCDRm (slave 2) + 1) × count clock cycle

Duty width 2 = (TAUDnCDRm (slave 3)) × count clock cycle

However, the delay width shall be set within the following range:
$0000_H \leq$ TAUDnCDRm (slave 2) < TAUDnCDRm (master)

**NOTES**

1. The waveform of TAUDTTOUTm (slave 3) becomes the waveform made by delaying the waveform of TAUDTTOUTm (slave 1) by the quantity generated by slave 2. It is impossible to make a delay longer than the pulse cycle.
2. If INTTAUD0Im of slave 2 is generated while slave 3 is counting, slave 3 restarts operation. Therefore, the waveform of TAUDTTOUTm (slave 3) is retained on the active level. In this case, TAUDTTOUTm (Slave-CH-3) cannot output the waveform generated by delaying the basic pulse of TAUDTTOUTm (Slave-CH-1).

### 25.15.4.3 Block Diagram and General Timing Diagram

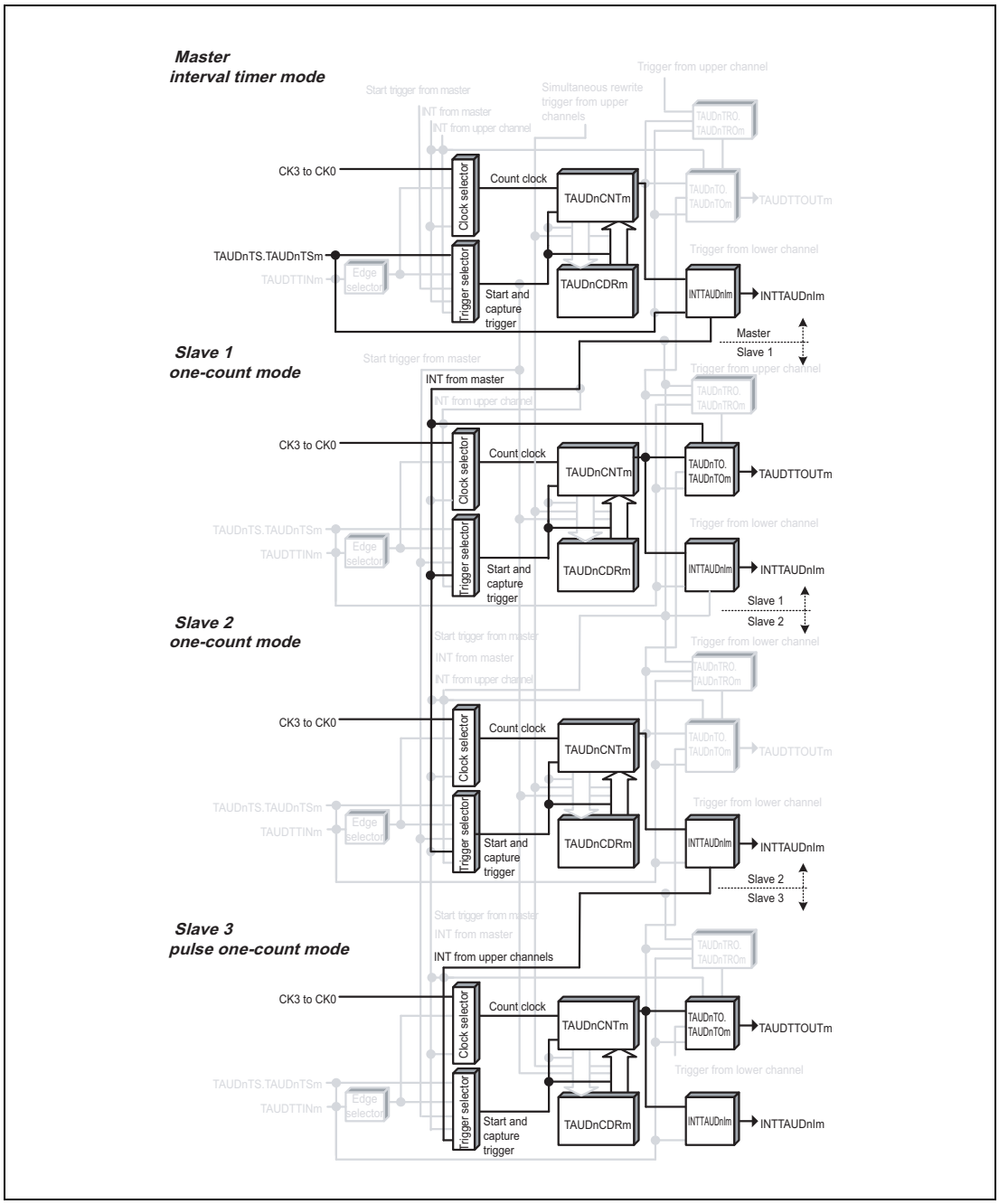



Figure 25.106 Block Diagram of Delay Pulse Output Function

The following settings apply to the general timing diagram.

- Slave channel 1: Positive logic (TAUDnTOL.TAUDnTOLm = 0)
- Slave channel 3: Positive logic (TAUDnTOL.TAUDnTOLm = 0)

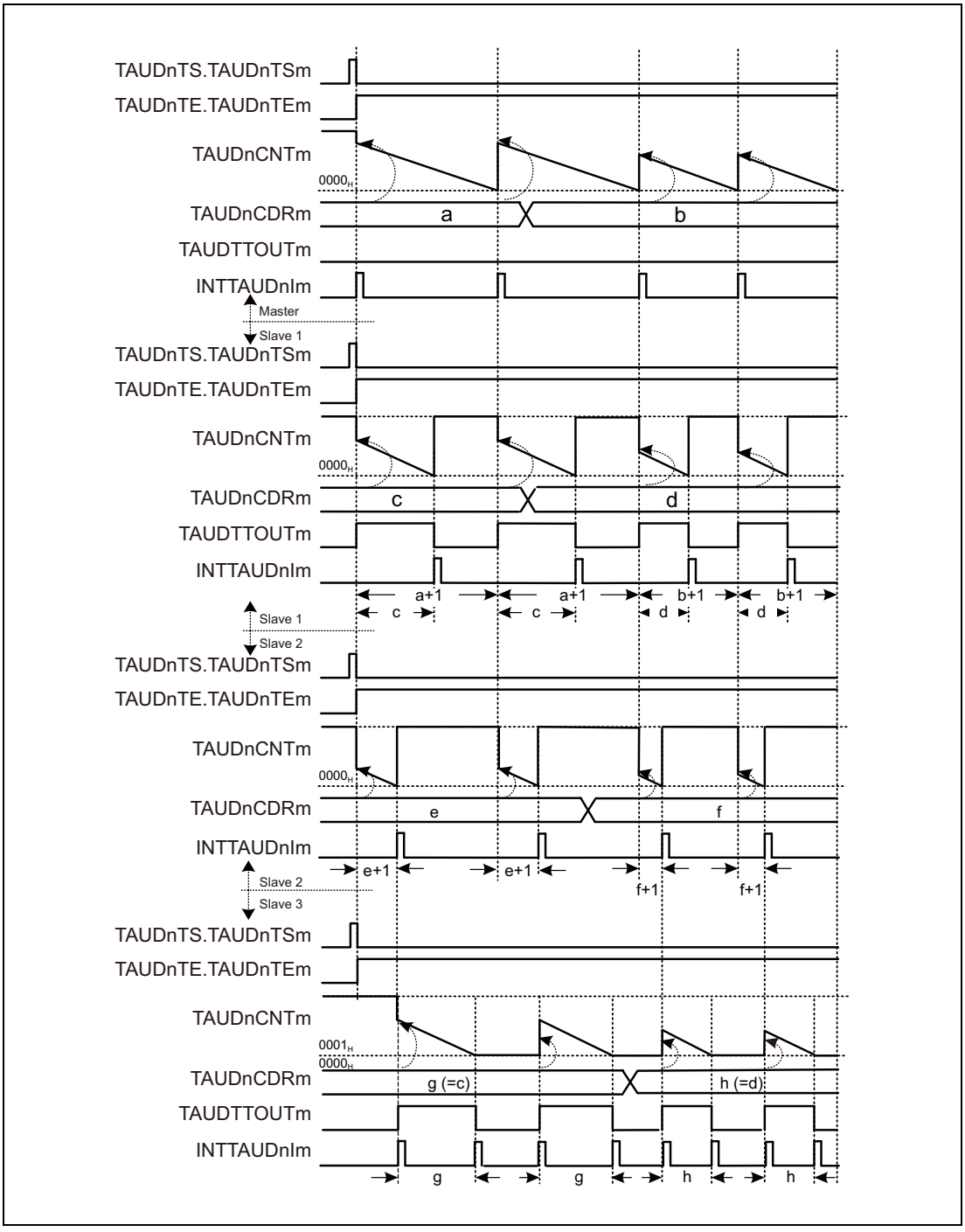



**Figure 25.107 General Timing Diagram of Delay Pulse Output Function**

**NOTE**

TAUDTTOUTm of slave channel 1 rises with a delay of one clock count after the rise of INTTAUDnIm of the master channel.

### 25.15.4.4 Register Settings for the Master Channel

#### (1) TAUDnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.166 Contents of the TAUDnCMORm Register for the Master Channel of the Delay Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 1: Master channel |
| 10 to 8 | TAUDnSTS[2:0] | 000: Trigger the counter using software. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0000: Interval timer mode |
| 0 | TAUDnMD0 | 1: INTTAUDnIm generated at the beginning of operation. |

#### (2) TAUDnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.167 Contents of the TAUDnCMURm Register for the Master Channel of the Delay Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for the master channel

TAUDnTOE.TAUDnTOEm is set to 0 because channel output mode is not used for the master channel with this function.

### (4) Simultaneous rewrite for the master channel

Both the master and slave channels should have the same simultaneous rewrite settings.

**Table 25.168 Simultaneous Rewrite Settings for the Master Channel of Delay Pulse Output Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Master channel is simultaneous rewrite control channel. |
| TAUDnRDM.TAUDnRDMm | 0: Generates a simultaneous rewrite trigger signal when the master channel starts to count. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

### 25.15.4.5 Register Settings for Slave Channel 1

#### (1) TAUDnCMORm for slave channel 1

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.169 Contents of the TAUDnCMORm Register for Slave Channel 1 of the Delay Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 100: INTTAUDnIm of master channel is a start trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0100: One-count mode |
| 0 | TAUDnMD0 | 1: Valid start trigger during operation |

#### (2) TAUDnCMURm for slave channel 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.170 Contents of the TAUDnCMURm Register for Slave Channel 1 of the Delay Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for slave channel 1

Table 25.171 Control Bit Settings for Slave Channel 1 in Synchronous Channel Output Mode 1

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 1: Synchronous channel operation |
| TAUDnTOC.TAUDnTOCm | 0: Operating mode 1 |
| TAUDnTOL.TAUDnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set this bit to 0 |
| TAUDnTRC.TAUDnTRCm | 0: Disables the operation as a real-time output trigger channel |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

### (4) Simultaneous rewrite for slave channel 1

Both the master and slave channels should have the same simultaneous rewrite settings.

Table 25.172 Simultaneous Rewrite Settings for Slave Channel 1 of Delay Pulse Output Function

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Master channel is simultaneous rewrite control channel. |
| TAUDnRDM.TAUDnRDMm | 0: Generates a simultaneous rewrite trigger signal when the master channel starts to count. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

### 25.15.4.6 Register Settings for Slave Channel 2

#### (1) TAUDnCMORm for slave channel 2

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.173 Contents of the TAUDnCMORm Register for Slave Channel 2 of the Delay Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 100: INTTAUDnIm of master channel is a start trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0100: One-count mode |
| 0 | TAUDnMD0 | 1: Valid start trigger during operation |

#### (2) TAUDnCMURm for slave channel 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.174 Contents of the TAUDnCMURm Register for Slave Channel 2 of the Delay Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for slave channel 2

TAUDnTOE.TAUDnTOEm is set to 0 because channel output mode is not used with this function.

### (4) Simultaneous rewrite for slave channel 2

Both the master and slave channels should have the same simultaneous rewrite settings.

**Table 25.175 Simultaneous Rewrite Settings for Slave Channel 2 of Delay Pulse Output Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Master channel is simultaneous rewrite control channel. |
| TAUDnRDM.TAUDnRDMm | 0: Generates a simultaneous rewrite trigger signal when the master channel starts to count. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

### 25.15.4.7 Register Settings for Slave Channel 3

#### (1) TAUDnCMORm for slave channel 3

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDnMAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDnMD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.176 Contents of the TAUDnCMORm Register for Slave Channel 3 of the Delay Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 101: INTTAUDnIm of upper channel (m - 1) is a start trigger regardless of master setting. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 1010: Pulse one-count mode |
| 0 | TAUDnMD0 | 1: Valid start trigger during operation |

#### (2) TAUDnCMURm for slave channel 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.177 Contents of the TAUDnCMURm Register for Slave Channel 3 of the Delay Pulse Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for slave channel 3

Table 25.178 Control Bit Settings in Independent Channel Output Mode 2

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 0: Independent channel output |
| TAUDnTOC.TAUDnTOCm | 1: Operating mode 2 |
| TAUDnTOL.TAUDnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set this bit to 0 |
| TAUDnTRC.TAUDnTRCm | 0: Disables the operation as a real-time output trigger channel |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

### (4) Simultaneous rewrite for slave channel 3

Both the master and slave channels should have the same simultaneous rewrite settings.

Table 25.179 Simultaneous Rewrite Settings for Slave Channel 3 of Delay Pulse Output Function

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Master channel is simultaneous rewrite control channel. |
| TAUDnRDM.TAUDnRDMm | 0: Generates a simultaneous rewrite trigger signal when the master channel starts to count. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

### 25.15.4.8 Operating Procedure for Delay Pulse Output Function

Table 25.180 Operating Procedure for Delay Pulse Output Function (1/2)

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Master channel: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.4.4, Register Settings for the Master Channel**.<br><br>Slave channel 1: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.4.5, Register Settings for Slave Channel 1**.<br><br>Slave channel 2: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.4.6, Register Settings for Slave Channel 2**.<br><br>Slave channel 3: Set the TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.4.7, Register Settings for Slave Channel 3**.<br><br>Set the value of TAUDnCDRm register of every channel. | Channel operation is stopped. |

**Table 25.180 Operating Procedure for Delay Pulse Output Function (2/2)**

| | Operation | TAUDn Status |
|---|---|---|
| Start Operation | Set TAUDnTS.TAUDnTSm of master and slave channels to 1 simultaneously.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm (master and slave channels) is set to 1 and the counters of master channel and slave channels 1 and 2 start.<br>INTTAUDnIm is generated on the master channel and TAUDTTOUTm (slave channel 1) is set. |
| During Operation | TAUDnCDRm can be changed at any time.<br>TAUDnCNTm and TAUDnRSF.TAUDnRSFm can be read at any time.<br><br>TAUDnRDT.TAUDnRDTm can be changed during operation. | TAUDnCDRm value of master channel and slave channels 1 and 2 is loaded to TAUDnCNTm and count down.<br><br>When the counter of master channel reaches $0000_H$:<br>• INTTAUDnIm (master) is generated.<br>• TAUDnCDRm value is reloaded into TAUDnCNTm (master) to continue count operation.<br>• TAUDnCDRm value is reloaded into TAUDnCNTm (slave 1/2) to count down.<br>• TAUDTTOUTm (slave 1) is set.<br><br>When TAUDnCNTm (slave 1) reaches $0000_H$:<br>• INTTAUDnIm (slave 1) is generated.<br>• TAUDTTOUTm (slave 1) is reset.<br><br>When TAUDnCNTm (slave 2) reaches $0000_H$:<br>• INTTAUDnIm (slave 2) is generated.<br>• INTTAUDnIm (slave 3) is generated.<br>• TAUDTTOUTm (slave 3) is set.<br>• TAUDnCDRm value is reloaded into TAUDnCNTm (slave 3) to count down operation.<br><br>When TAUDnCNTm (slave 3) reaches $0001_H$:<br>• INTTAUDnIm (slave 3) is generated.<br>• TAUDTTOUTm (slave 3) is reset. |
| Stop Operation | Set TAUDnTT.TAUDnTTm of master and slave channels to 1 simultaneously.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm and TAUDTTOUTm stop and retain their current values. |

Restart Operation

### 25.15.4.9 Specific Timing Diagrams

#### (1) Duty cycle (slave 3) = 100%

The following values apply to **Figure 25.108**:

- TAUDnCDRm (master) = $000A_H$
- TAUDnCDRm (slave 1) = $000B_H$
- TAUDnCDRm (slave 2) = $0000_H$
- TAUDnCDRm (slave 3) = $000B_H$

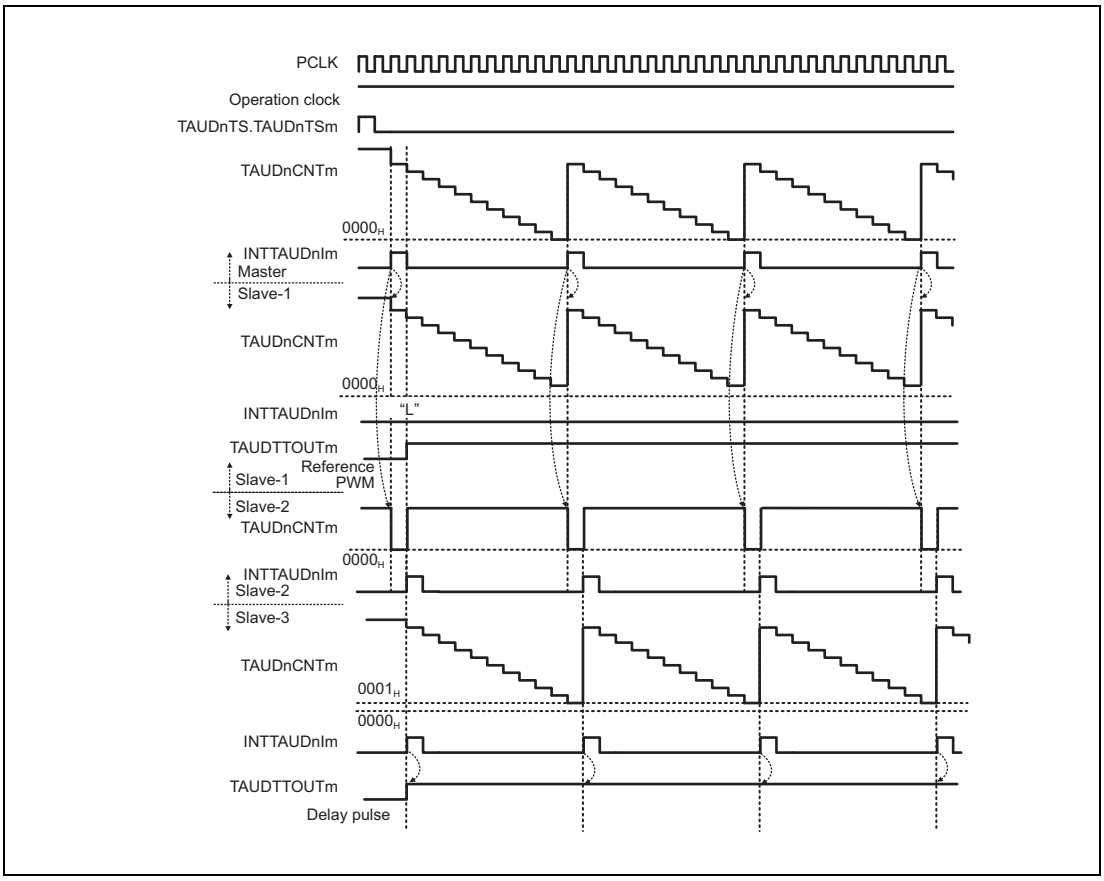



**Figure 25.108 Duty Cycle (Slave 3) = 100%**

- If the value of TAUDnCDRm (slaves 1 and 3) is higher than the value of TAUDnCDRm (master), the counter of the slave channel 1 cannot reach $0000_H$ and cannot generate interrupts. TAUDTTOUTm of channels 1 and 3 remain in the active state.

### (2) TAUDTTOUTm (slave 1) = TAUDTTOUTm (slave 3)

The following values apply to **Figure 25.109**.

- TAUDnCDRm (master) = $000A_H$
- TAUDnCDRm (slave 1) = $0005_H$
- TAUDnCDRm (slave 2) = $0000_H$
- TAUDnCDRm (slave 3) = $0005_H$

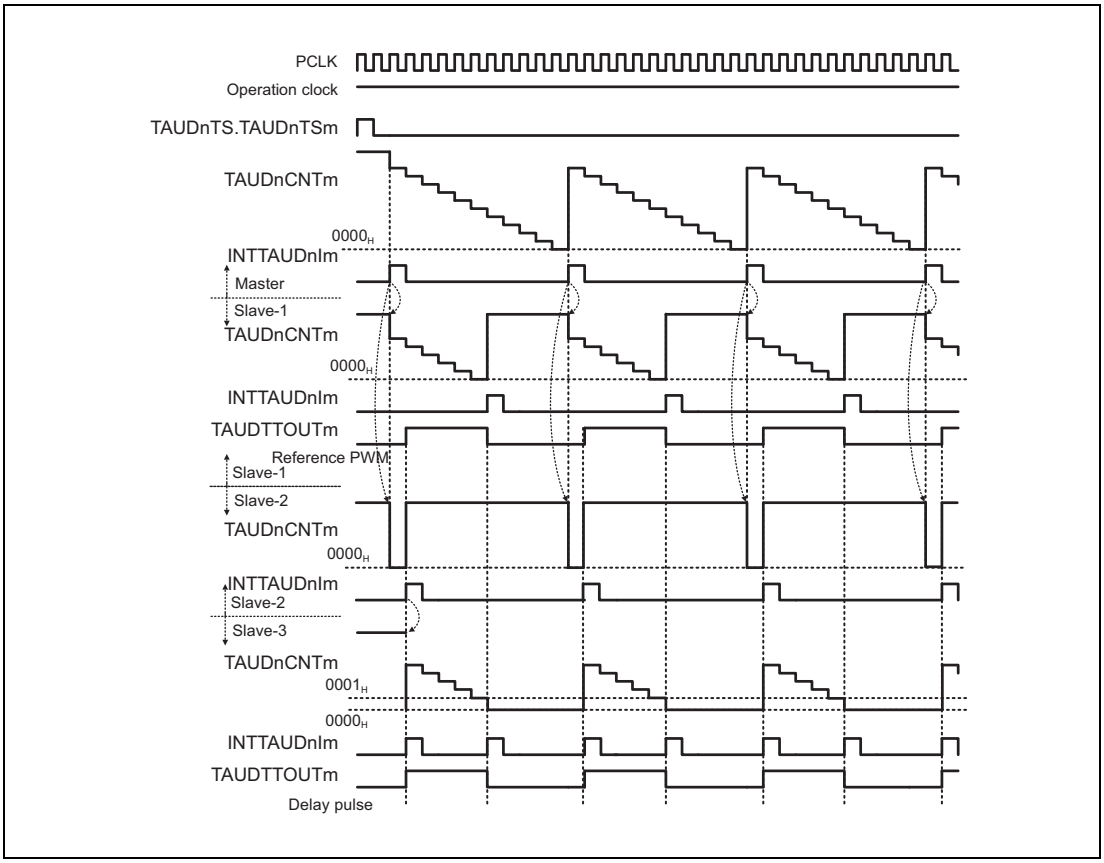



**Figure 25.109 TAUDTTOUTm (Slave 1) = TAUDTTOUTm (Slave 3)**

- If TAUDnCDRm (slave 2) = $0000_H$, the counter of slave channel 3 starts counting one count clock later than the counter of slave channel 1. The reference pulse and the delay pulse are output with a delay of one clock count.

## 25.15.5 Offset Trigger Output Function

### 25.15.5.1 Overview

#### Summary

This function generates a PWM output using a master channel and a slave channel, enabling the pulse width (duration) of the TAUDTTOUTm to be set. The pulse cycle is set by detecting a valid input edge of master channel. The pulse width is specified on the slave channel.

#### Prerequisites

- Two channels
- The operating mode for the master channel should be set to capture mode. (See **Table 25.181, Contents of the TAUDnCMORm Register for the Master Channel of the Offset Trigger Output Function**.)
- The operating mode for slave channels should be set to one-count mode. (See **Table 25.184, Contents of the TAUDnCMORm Register for the Slave Channel of the Offset Trigger Output Function**.)
- The output mode for slave channels should be set to synchronous channel output mode 1. (See **Section 25.7, Channel Output Modes**.)
- TAUDTTOUTm is not used with the master channel of this function.

#### Functional description

The counter can be enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This makes TAUDnTE.TAUDnTEm = 1, enabling count operation. The master channel counter (TAUDnCNTm) starts to count up from $0000_H$.

- Master channel:
  When a valid TAUDTTINm input edge is detected, the current value of the counter (TAUDnCNTm) is loaded into the data register of master channel (TAUDnCDRm). INTTAUDnIm is generated and the counter restarts to count up from $0000_H$.
- Slave channel:
  If INTTAUDnIm is generated on the master channel, the TAUDTTOUTm (slave) signal is set and the counter of the slave channel is triggered. The current value of TAUDnCDRm (slave) is loaded into TAUDnCNTm (slave) and the counter starts to count down from this value.
  When the counter reaches $0000_H$ (duty time has elapsed), INTTAUDnIm is generated and TAUDTTOUTm signal is reset. The counter returns to $FFFF_H$ and awaits the next INTTAUDnIm of the master channel.

The counter can be stopped by setting TAUDnTT.TAUDnTTm of master and slave channels to 1. This sets TAUDnTE.TAUDnTEm to 0. TAUDnCNTm and TAUDTTOUTm of master and slave channels stop but retain their values. The counters can be restarted by setting TAUDnTS.TAUDnTSm to 1.

### 25.15.5.2 Equations

Pulse width = (TAUDnCDRm (slave)) × count clock cycle

Duty cycle [%] = [TAUDnCDRm (slave)/(TAUDnCDRm (master) + 1)] × 100

– Duty cycle = 0%
  TAUDnCDRm (slave) = $0000_H$
– Duty cycle = 100%
  TAUDnCDRm (slave) ≥ TAUDnCDRm (master) + 1

### 25.15.5.3 Block Diagram and General Timing Diagram

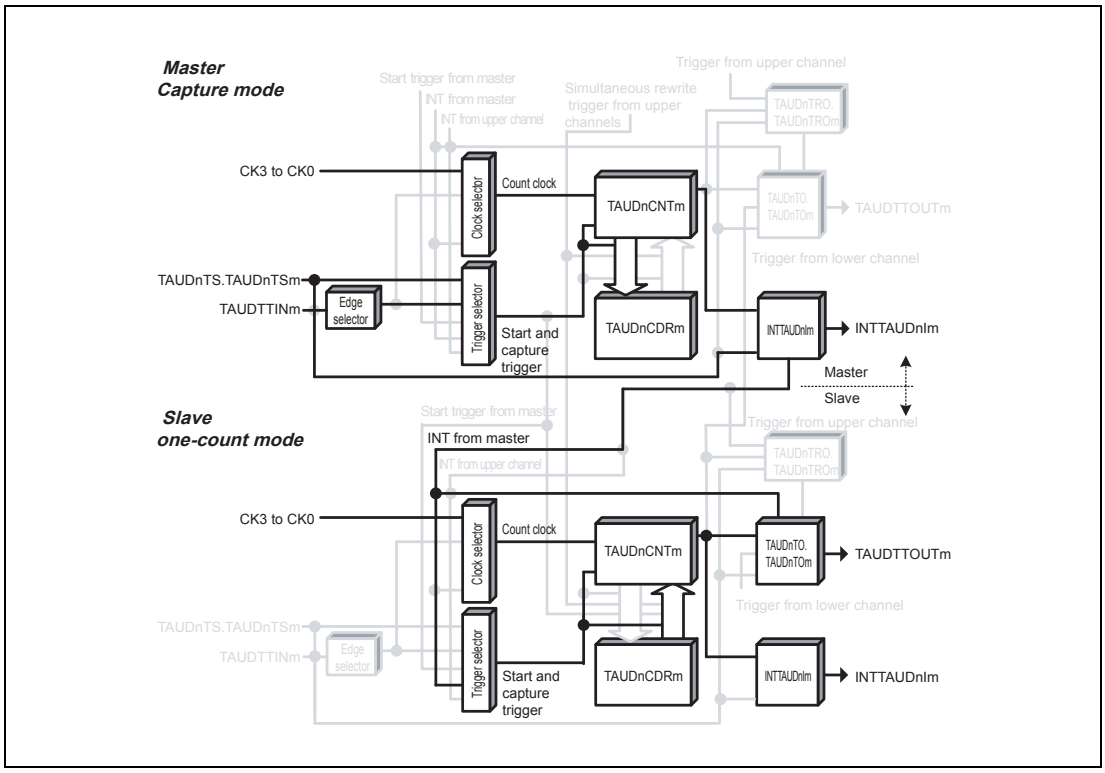



**Figure 25.110 Block Diagram of Offset Trigger Output Function**

The following settings apply to the general timing diagram.

- Detection of falling edge (TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)

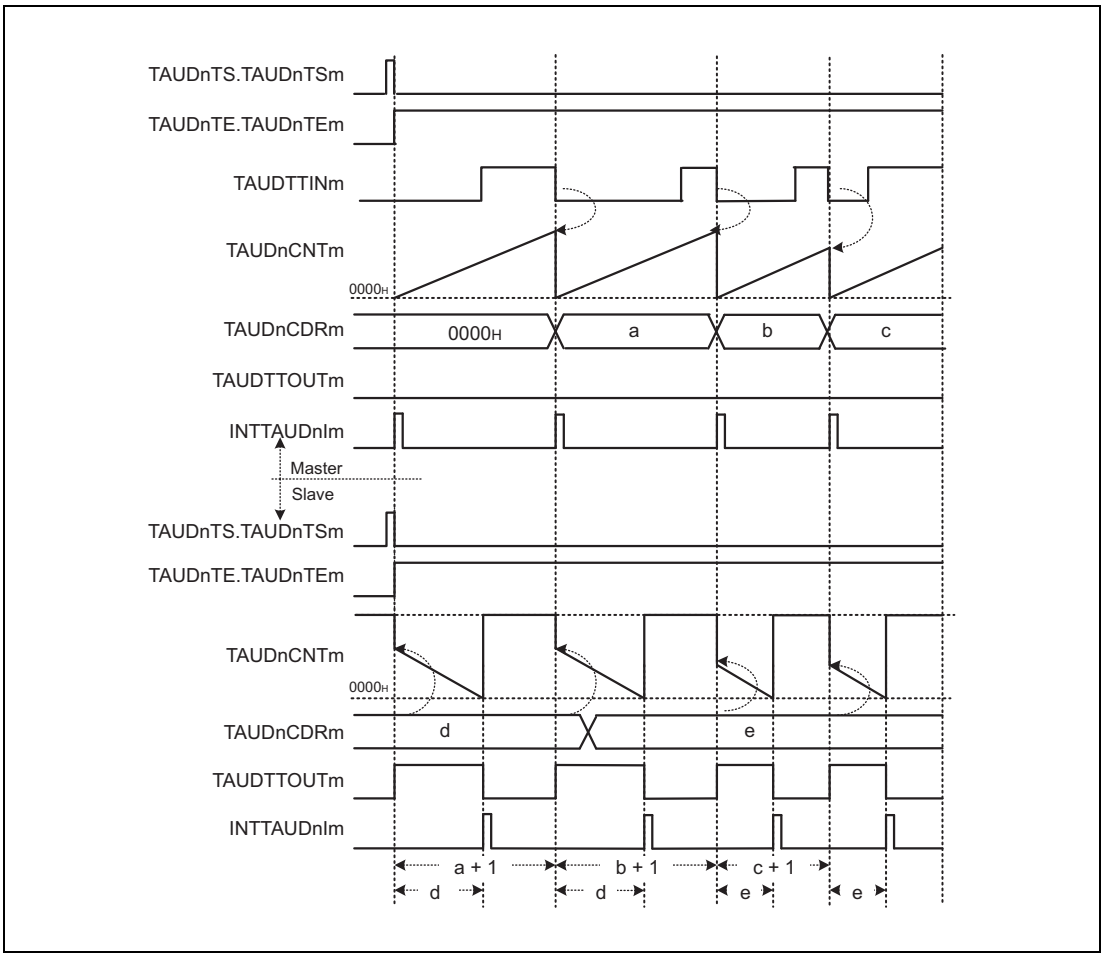



**Figure 25.111 General Timing Diagram of Offset Trigger Output Function**

**NOTE**

TAUDTTOUTm of the slave channel rises with a delay of one clock count after the rise of INTTAUDnIm of the master channel.

### 25.15.5.4 Register Settings for the Master Channel

#### (1) TAUDnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDnMAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDnMD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.181 Contents of the TAUDnCMORm Register for the Master Channel of the Offset Trigger Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 1: Master channel |
| 10 to 8 | TAUDnSTS[2:0] | 001: Valid TAUDTTINm input edge signal is used as the start trigger |
| 7, 6 | TAUDnCOS[1:0] | 11: Updated upon detection of a valid TAUDTTINm input edge or when a counter overflow occurs:<br>– Detection of valid TAUDTTINm input edge: The counter value is written into TAUDnCDRm.<br>– Occurrence of overflow: $FFFF_H$ is written into TAUDnCDRm. A valid TAUDTTINm input edge to be detected next is ignored.<br>TAUDnCSRm.TAUDnOVF is set when a counter overflow occurs, and cleared by setting TAUDnCSCm.TAUDnCLOV = 1. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0010: Capture mode |
| 0 | TAUDnMD0 | 1: INTTAUDnIm generated at the beginning of operation. |

#### (2) TAUDnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.182 Contents of the TAUDnCMURm Register for the Master Channel of the Offset Trigger Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Detection of falling edge<br>01: Detection of rising edge<br>10: Detection of rising and falling edges<br>11: Setting prohibited |

### (3) Channel output mode for the master channel

TAUDnTOE.TAUDnTOEm is set to 0 because channel output mode is not used with this function.

### (4) Simultaneous rewrite for the master channel

Simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the offset trigger output function. Therefore, these registers should be set to 0.

**Table 25.183 Simultaneous Rewrite Settings for the Master Channel of Offset Trigger Output Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

### 25.15.5.5 Register Settings for Slave Channels

#### (1) TAUDnCMORm for slave channels

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.184 Contents of the TAUDnCMORm Register for the Slave Channel of the Offset Trigger Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 100: INTTAUDnIm of master channel is a start trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0100: One-count mode |
| 0 | TAUDnMD0 | 1: Enables start trigger detection while counting. |

#### (2) TAUDnCMURm for slave channels

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.185 Contents of the TAUDnCMURm Register for the Slave Channel of the Offset Trigger Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for slave channels

Table 25.186 Control Bit Settings in Synchronous Channel Output Mode 1

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 1: Synchronous channel operation |
| TAUDnTOC.TAUDnTOCm | 0: Operating mode 1 |
| TAUDnTOL.TAUDnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set this bit to 0 |
| TAUDnTRC.TAUDnTRCm | 0: Disables the operation as a real-time output trigger channel |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

### (4) Simultaneous rewrite for slave channels

Simultaneous rewrite registers (TAUDnRDE, TAUDnRDS, TAUDnRDM, and TAUDnRDC) cannot be used with the offset trigger output function. Therefore, these registers should be set to 0.

Table 25.187 Simultaneous Rewrite Settings for Slave Channels of Offset Trigger Output Function

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 0: Disables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | When simultaneous rewrite is disabled (TAUDnRDE.TAUDnRDEm = 0), set these bits to 0 |
| TAUDnRDM.TAUDnRDMm | |
| TAUDnRDC.TAUDnRDCm | |

### 25.15.5.6 Operating Procedure for Offset Trigger Output Function

Table 25.188 Operating Procedure for Offset Trigger Output Function

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Master channel: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.5.4, Register Settings for the Master Channel**.<br><br>Slave channel: Set TAUDnCMORm and TAUDnCMURm registers and channel output mode as described in **Section 25.15.5.5, Register Settings for Slave Channels**.<br><br>The TAUDnCDRm register of master channel functions as a capture register.<br>Set the value of TAUDnCDRm register of slave channel. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm of master and slave channels to 1 simultaneously.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm (master and slave channels) is set to 1 and the counters of master and slave channels start:<br>• TAUDnCNTm (master) counts up.<br>• TAUDnCDRm value is loaded into TAUDnCNTm (slave) to perform counting down.<br>INTTAUDnIm is generated on the master channel and TAUDTTOUTm (slave) is set. |
| During Operation | TAUDnCDRm can be changed at any time.<br><br>TAUDnCSCm.TAUDnCLOV can be set to 1.<br>TAUDnCDRm of slave channel can be changed after the generation of INTTAUDnIm (master).<br>TAUDnCNTm and TAUDnCSRm can be read at any time. | When TAUDnCNTm of the slave = $0000_H$:<br>• INTTAUDnIm (slave) is generated.<br>• TAUDTTOUTm (slave) is reset, and the counter of slave channel stops.<br><br>When TAUDTTINm input edge is detected on the master channel:<br>• INTTAUDnIm (master) is generated.<br>• TAUDnCNTm (master) is reset to $0000_H$ and then continues count operation subsequently.<br>• TAUDnCDRm value is reloaded into TAUDnCNTm (slave) to perform counting down.<br>• TAUDTTOUTm (slave) is set. |
| Stop Operation | Set TAUDnTT.TAUDnTTm of master and slave channels to 1 simultaneously.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm and TAUDTTOUTm stop and retain their current values. |

Restart Operation

### 25.15.5.7 Specific Timing Diagrams

#### (1) Duty cycle = 0%

The following settings apply to this diagram.

- Detection of falling edge (TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)

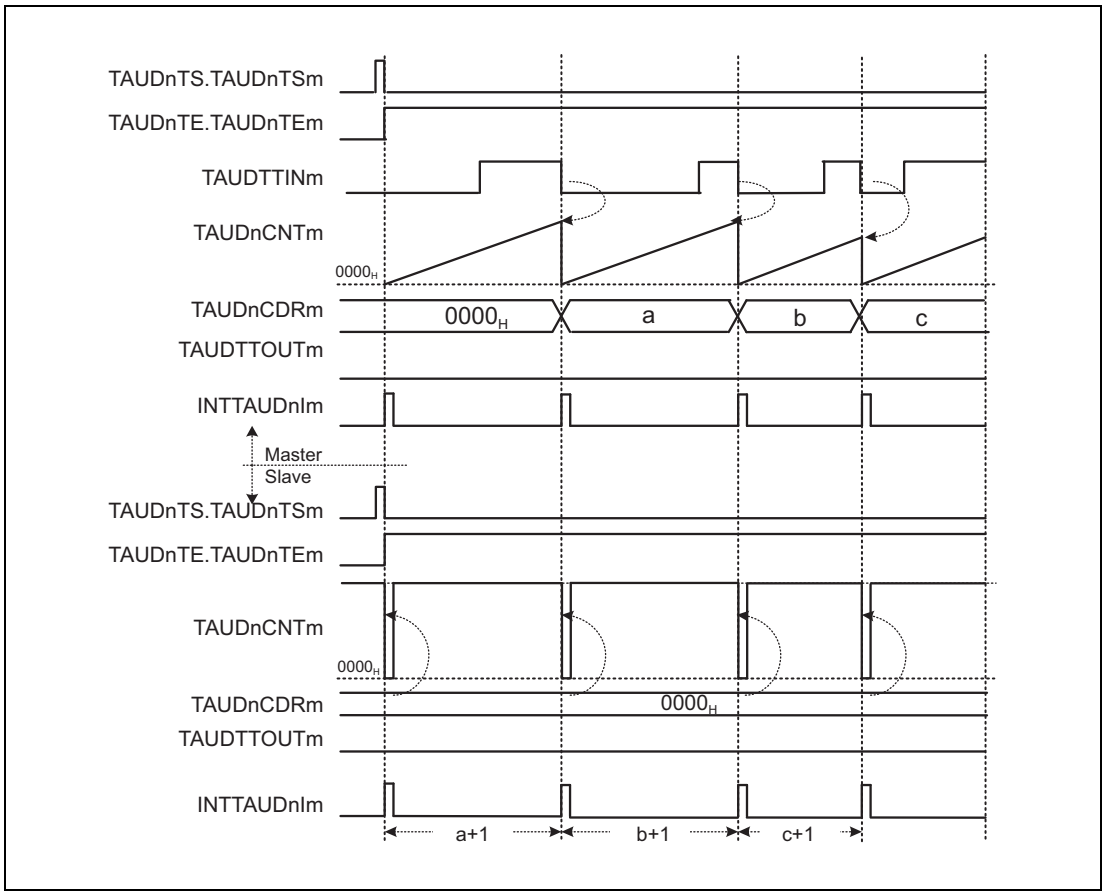



**Figure 25.112 TAUDnCDRm (Slave) = $0000_H$**

- When TAUDnCDRm (slave) = $0000_H$, $0000_H$ is written to TAUDnCNTm every time the master channel generates an interrupt (INTTAUDnIm), and TAUDnCNTm cannot start to count. The TAUDTTOUTm remains inactive.
- TAUDnCNTm (slave) generates an interrupt every time the value of TAUDnCDRm is reloaded. The slave and the master channels generate interrupts in the same cycle.

### (2) Duty cycle = 100%

The following settings apply to this diagram.

- Detection of falling edge (TAUDnCMURm.TAUDnTIS[1:0] = $00_B$)

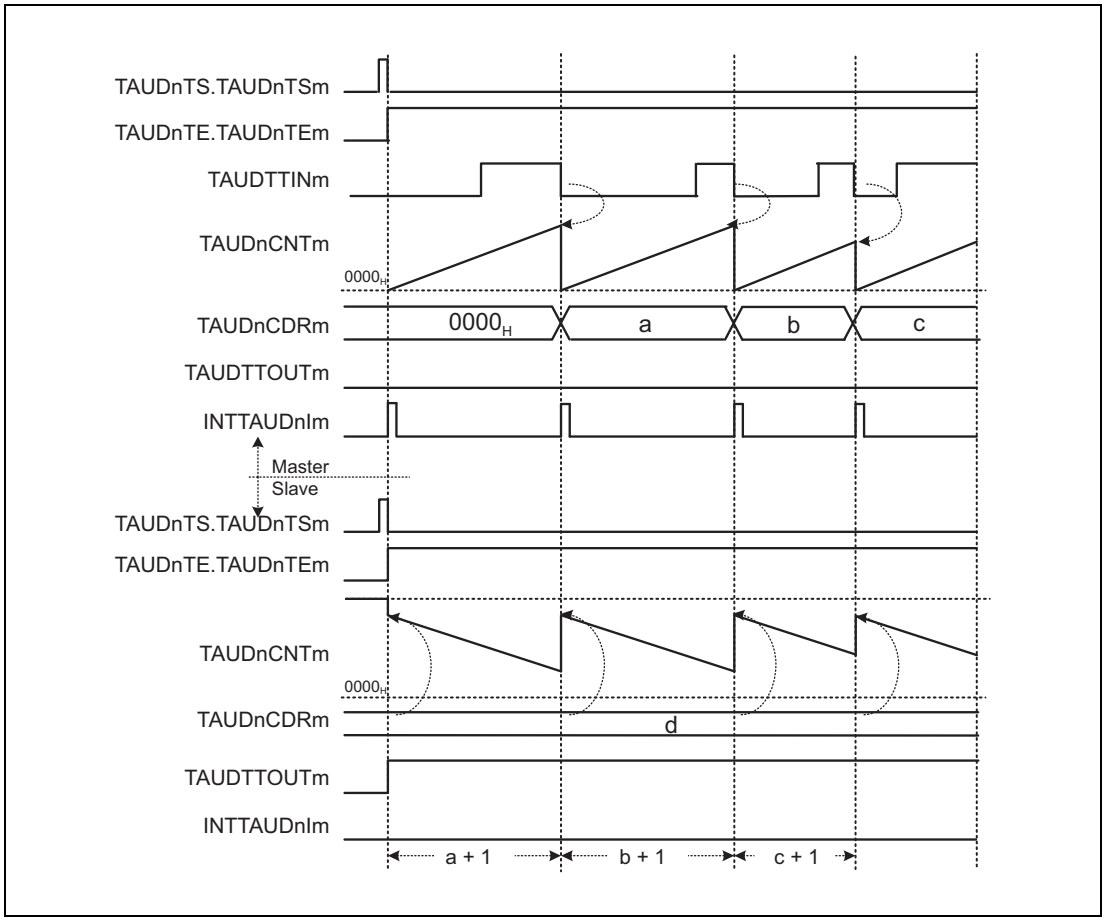



**Figure 25.113 TAUDnCDRm (Slave) ≥ TAUDnCDRm (Master) + 1**

- If the value TAUDnCDRm (slave) is higher than the interval of valid input edges, the counter of the slave channel cannot reach $0000_H$ and cannot generate interrupts. The TAUDTTOUTm remains at active state.

## 25.15.6 A/D Conversion Trigger Output Function Type 1

### 25.15.6.1 Overview

#### Summary

This function is identical to **Section 25.15.1, PWM Output Function**, except that TAUDTTOUTm is not output.

This is achieved by setting the channel output mode for the slave to independent channel output mode controlled by software.

### 25.15.6.2 Block Diagram and General Timing Diagram

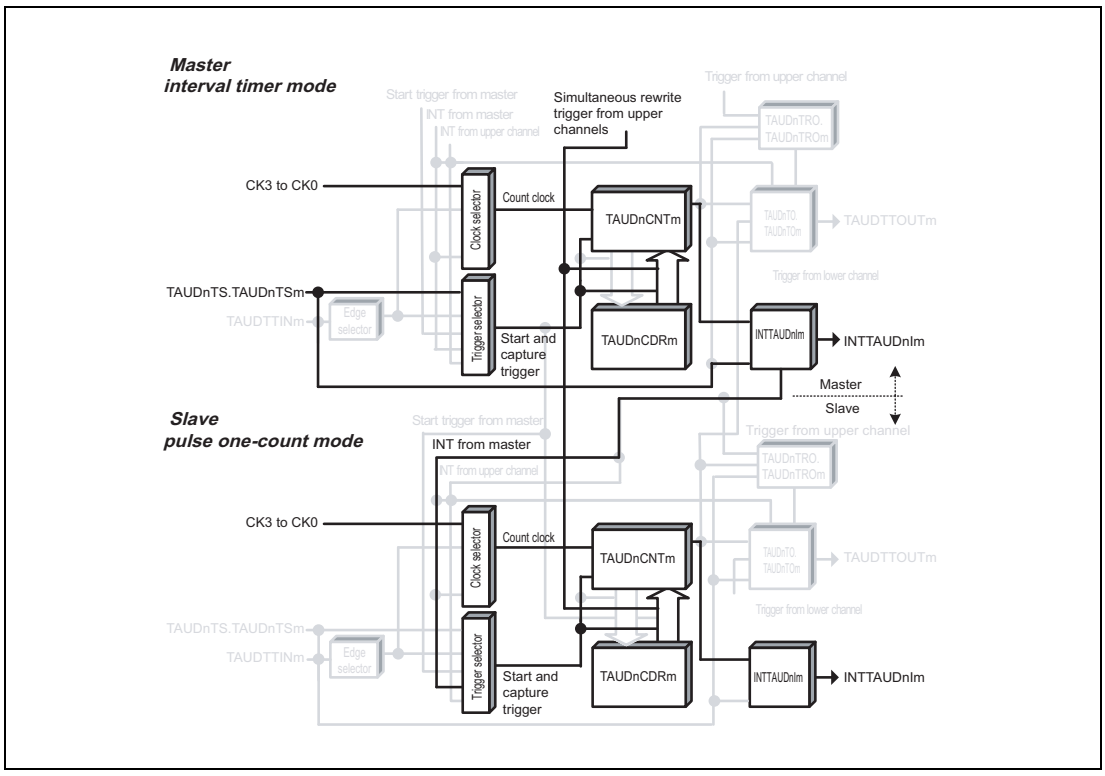



**Figure 25.114 Block Diagram of A/D Conversion Trigger Output Function Type 1**

The following settings apply to the general timing diagram.

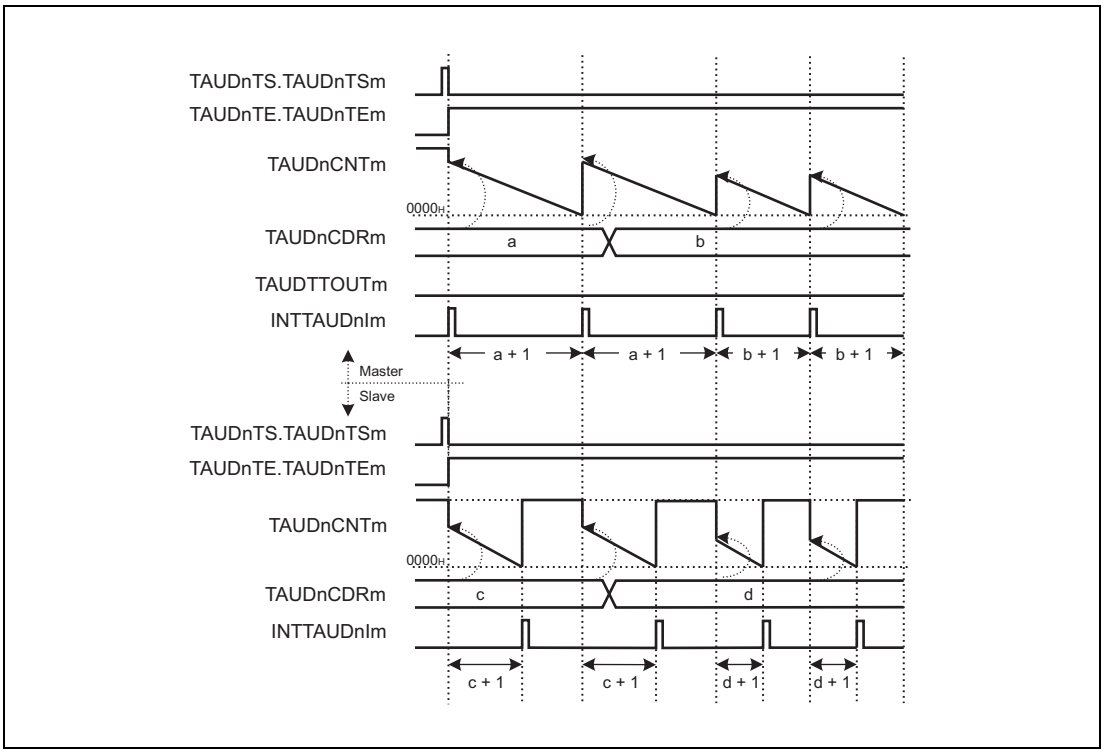



**Figure 25.115 General Timing Diagram of A/D Conversion Trigger Output Function Type 1**

## 25.15.7 Triangle PWM Output Function

### 25.15.7.1 Overview

#### Summary

This function generates multiple triangle PWM outputs by using a master and one or more slave channels. It enables the pulse cycle (frequency) and the duty cycle of TAUDTTOUTm to be set using the master and slave channels respectively.

The master channel generates a carrier cycle. The first cycle of the master channel controls the down status and the second cycle controls the up status of the slave counter.

#### Prerequisites

- Two channels
- The operating mode for the master channels should be set to interval timer mode. (See **Table 25.189, Contents of the TAUDnCMORm Register for the Master Channel of the Triangle PWM Output Function**.)
- The operating mode for slave channels should be set to count-up/-down mode. (See **Table 25.193, Contents of the TAUDnCMORm Register for the Slave Channel of the Triangle PWM Output Function**.)
- The channel output mode for the master channel should be set to independent channel output mode 1. (See **Section 25.7, Channel Output Modes**.)
- The channel output mode for slave channels should be set to synchronous channel output mode 2. (See **Section 25.7, Channel Output Modes**.)
- The following settings allow the TAUDTTOUTm signal to be at high level during the down status of a carrier cycle.
  - If TAUDnCMORm.TAUDnMD0 (master) bit is set to 0, TAUDnTO.TAUDnTOm should be set to 1 while TAUDnTOE.TAUDnTOEm is set to 0 (recommended setting).
  - If TAUDnCMORm.TAUDnMD0 (master) bit is set to 1, TAUDnTO.TAUDnTOm should be set to 0 while TAUDnTOE.TAUDnTOEm is set to 0.

#### Functional description

The counters are enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1 for every channel. This in turn sets TAUDnTE.TAUDnTEm, enabling count operation. The current values of TAUDnCDRm (master and slave) are loaded into TAUDnCNTm (master and slave) and the counters start counting down from these values. When the TAUDnCMORm.TAUDnMD0 bit of master channel is set to 1, an interrupt is generated and TAUDTTOUTm signal of master toggles.

- Master channel:
  When the counter of master channel reaches $0000_H$ (pulse cycle time has elapsed), INTTAUDnIm is generated and the TAUDTTOUTm signal toggles. TAUDnCNTm then reloads the TAUDnCDRm value and counts down.

- Slave channel:
  If INTTAUDnIm is generated on the master channel, the counter of the slave channel is triggered.
  - If the slave counter is counting down, the count direction changes.
  - If the slave counter is counting up, the TAUDnCDRm value is reloaded and the counter starts to count down.

  When the counter of the slave channel reaches $0001_H$ while counting up or down, INTTAUDnIm is generated and the TAUDTTOUTm (slave) signal is set/reset.
  The counter continues count-up/-down and waits for the next INTTAUDnIm of the master channel.
  Setting TAUDnTOL.TAUDnTOLm allows TAUDTTOUTm signal switching between normal phase and reverse phase during operation.

The counter can be stopped by setting TAUDnTT.TAUDnTTm of master and slave channels to 1. This sets TAUDnTE.TAUDnTEm = 0. TAUDnCNTm and TAUDTTOUTm of master and slave channels stop but retain their values.

**Conditions**

This function enables simultaneous rewrite. See **Section 25.6, Simultaneous Rewrite**.

### 25.15.7.2 Equations

Pulse cycle = (TAUDnCDRm (master) + 1) × count clock cycle

$0000_H \leq$ TAUDnCDRm (master) $< FFFF_H$

Carrier cycle (down/up) = (TAUDnCDRm (master) + 1) × 2 × count clock cycle

Duty cycle 100 [%] =
[(TAUDnCDRm (master) + 1 – TAUDnCDRm (slave))/(TAUDnCDRm (master) + 1)] × 100

- Duty cycle = [%]
  TAUDnCDRm (slave) = $0000_H$
- Duty cycle = 0%
  TAUDnCDRm (slave) ≥ TAUDnCDRm (master) + 1

### 25.15.7.3 Block Diagram and General Timing Diagram

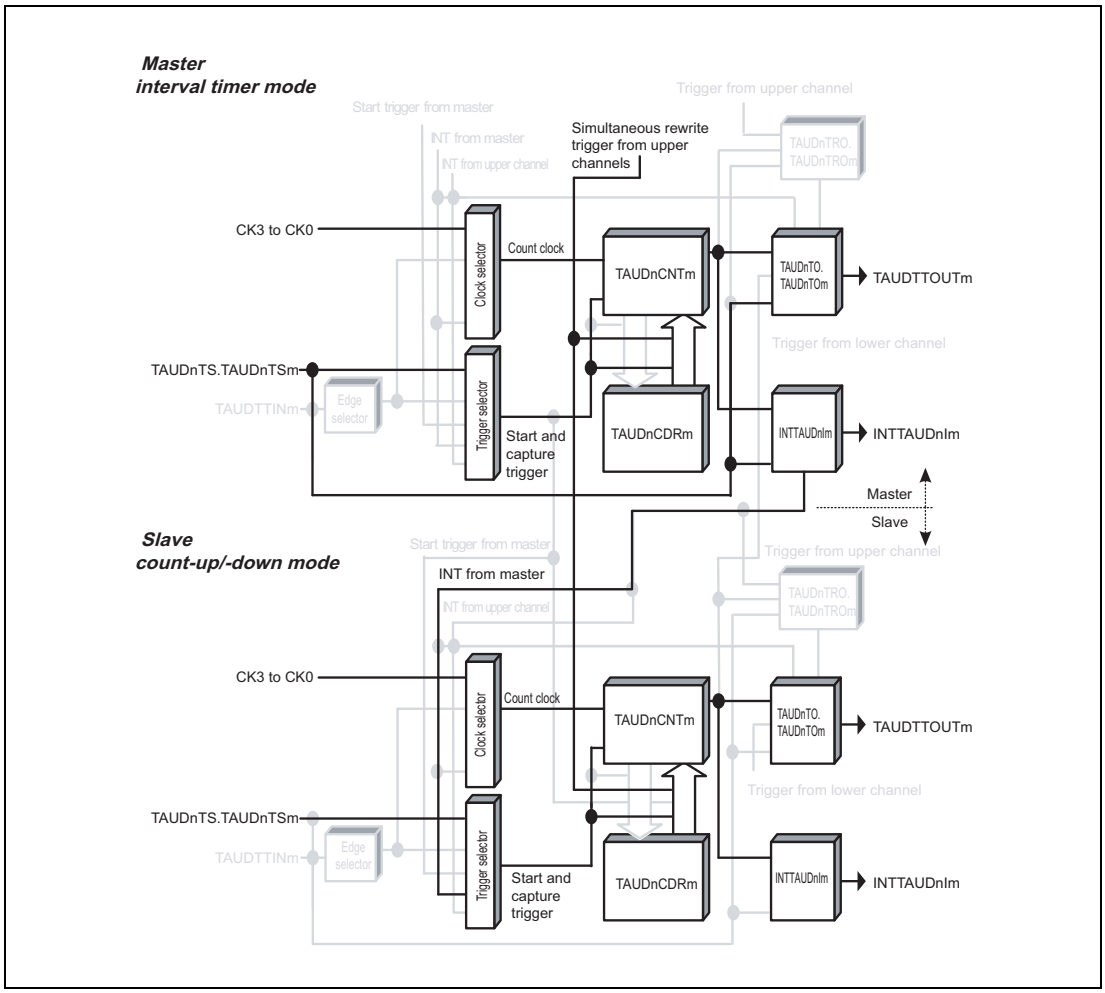



**Figure 25.116 Block Diagram of Triangle PWM Output Function**

The following settings apply to the general timing diagram.

- Master channel
  - INTTAUDnIm is generated at the beginning of operation.
    (TAUDnCMORm.TAUDnMD0 = 1)

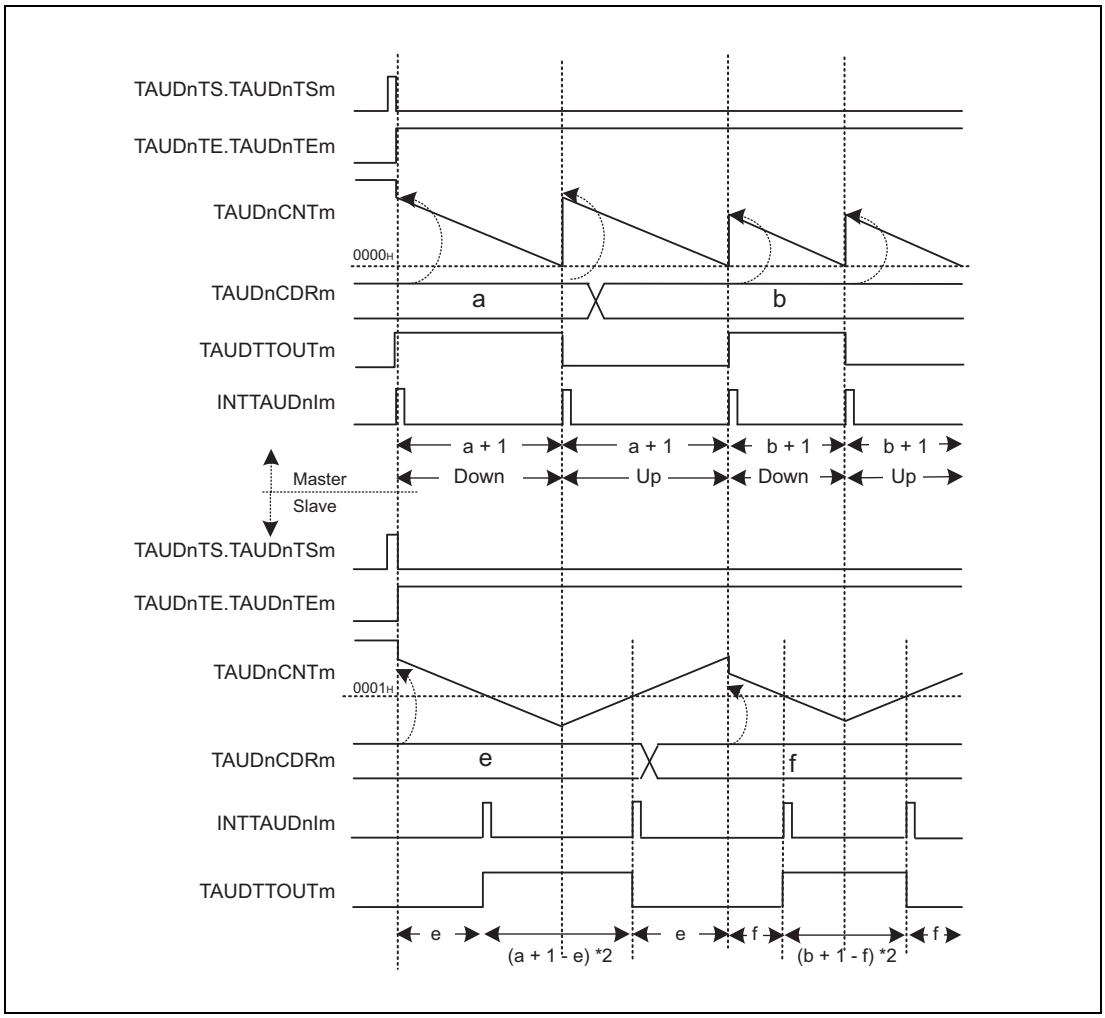



**Figure 25.117 General Timing Diagram of Triangle PWM Output Function**

### 25.15.7.4 Register Settings for the Master Channel

#### (1) TAUDnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.189 Contents of the TAUDnCMORm Register for the Master Channel of the Triangle PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 1: Master channel |
| 10 to 8 | TAUDnSTS[2:0] | 000: Trigger the counter using software. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0000: Interval timer mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm is not generated to toggle TAUDTTOUTm at the beginning of an operation.<br>1: INTTAUDnIm is generated and TAUDTTOUTm is toggled at the beginning of operation. |

#### (2) TAUDnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.190 Contents of the TAUDnCMURm Register for the Master Channel of the Triangle PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for the master channel

Table 25.191 Control Bit Settings in Independent Channel Output Mode 1

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 0: Independent channel output |
| TAUDnTOC.TAUDnTOCm | 0: Operating mode 1 (Toggle mode if TAUDnTOM.TAUDnTOMm = 0) |
| TAUDnTOL.TAUDnTOLm | 0: The setting is disabled in toggle mode (the value after reset). |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set this bit to 0 |
| TAUDnTRC.TAUDnTRCm | 0: Disables the operation as a real-time output trigger channel |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

### (4) Simultaneous rewrite for the master channel

Both the master and slave channels should have the same simultaneous rewrite settings.

Table 25.192 Simultaneous Rewrite Settings for the Master Channel of Triangle PWM Output Function

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Selects a master channel for simultaneous rewrite triggers.<br>1: Selects an upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 1: A simultaneous rewrite trigger signal is generated when master channel starts to count and the corresponding slave channel is at the peak of a triangular wave cycle. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

**NOTE**

If TAUDnRDS.TAUDnRDSm = 1, it is necessary for an upper channel higher than the master channel to generate a simultaneous rewrite trigger signal.

### 25.15.7.5 Register Settings for Slave Channels

#### (1) TAUDnCMORm for slave channels

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.193 Contents of the TAUDnCMORm Register for the Slave Channel of the Triangle PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 111: Up/down output trigger signal of master channel |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 1001: Count-up/-down mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm not generated at the beginning of operation. |

#### (2) TAUDnCMURm for slave channels

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.194 Contents of the TAUDnCMURm Register for the Slave Channel of the Triangle PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for slave channels

**Table 25.195 Control Bit Settings in Synchronous Channel Output Mode 2**

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 1: Synchronous channel operation |
| TAUDnTOC.TAUDnTOCm | 1: Operating mode 2 |
| TAUDnTOL.TAUDnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set this bit to 0 |
| TAUDnTRC.TAUDnTRCm | 0: Disables the operation as a real-time output trigger channel |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

### (4) Simultaneous rewrite for slave channels

Both the master and slave channels should have the same simultaneous rewrite settings.

**Table 25.196 Simultaneous Rewrite Settings for Slave Channels of Triangle PWM Output Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Selects a master channel for simultaneous rewrite triggers.<br>1: Selects an upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 1: A simultaneous rewrite trigger signal is generated when the master channel starts to count and the corresponding slave channel is at the peak of a triangular wave cycle. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

### 25.15.7.6 Operating Procedure for Triangle PWM Output Function

Table 25.197 Operating Procedure for Triangle PWM Output Function

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Master channel: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.7.4, Register Settings for the Master Channel**.<br><br>Slave channel: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.7.5, Register Settings for Slave Channels**.<br><br>Set the value of TAUDnCDRm register of every channel. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm of master and slave channels to 1 simultaneously.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm (master and slave channels) is set to 1 and the counters of master and slave channels start.<br>INTTAUDnIm (master) is generated on the master channel if TAUDnCMORm.TAUDnMD0 is set to 1. |
| During Operation | TAUDnCDRm can be changed at any time.<br>TAUDnTOL.TAUDnTOLm can be changed.<br>TAUDnCNTm and TAUDnRSF.TAUDnRSFm can be read at any time.<br><br>TAUDnRDT.TAUDnRDTm can be changed during operation. | TAUDnCDRm value of master and slave channels is loaded into TAUDnCNTm to count down. When the counter of master channel reaches $0000_H$:<br>• INTTAUDnIm (master) is generated.<br>• TAUDTTOUTm (master) is toggled.<br>• TAUDnCDRm value is reloaded into TAUDnCNTm (master) to continue count operation.<br>• TAUDnCDRm value is reloaded into TAUDnCNTm (slave) or counting is started in opposite direction.<br>When TAUDnCNTm of slave channel reaches $0001_H$:<br>• INTTAUDnIm (slave) is generated.<br>• TAUDTTOUTm (slave) is set in the count-down status or reset in count-up status. |
| Stop Operation | Set TAUDnTT.TAUDnTTm of master and slave channels to 1 simultaneously.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm and TAUDTTOUTm stop and retain their current values. |

Restart Operation

### 25.15.7.7 Specific Timing Diagrams

#### (1) Duty cycle = 0%

The following settings apply to the general timing diagram.

- Master channel:
  - INTTAUDnIm is generated at the beginning of operation. (TAUDnCMORm.TAUDnMD0 = 1)
  - TAUDnCDRm = a = $5_H$
- Slave channel:
  - TAUDnCDRm = $6_H$

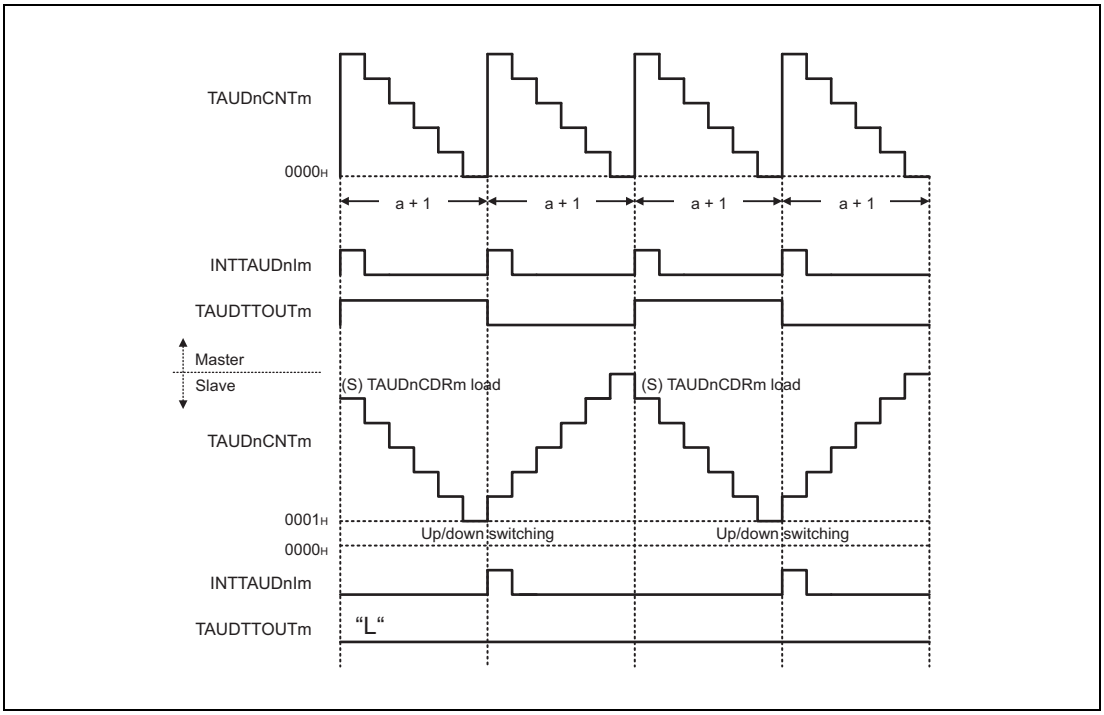



**Figure 25.118 TAUDnCDRm (Slave) ≥ TAUDnCDRm (Master) + 1**

- If TAUDnCDRm (slave) value ≥ TAUDnCDRm (master) value +1, INTTAUDnIm of the slave channel is not generated while counting down. TAUDTTOUTm remains low because there is no set signal to be detected.

### (2) Duty cycle = 100%

The following settings apply to the general timing diagram.

- Master channel:
  - INTTAUDnIm is generated at the beginning of operation. (TAUDnCMORm.TAUDnMD0 = 1)
  - TAUDnCDRm = a = $5_H$
- Slave channel:
  - TAUDnCDRm = $0_H$

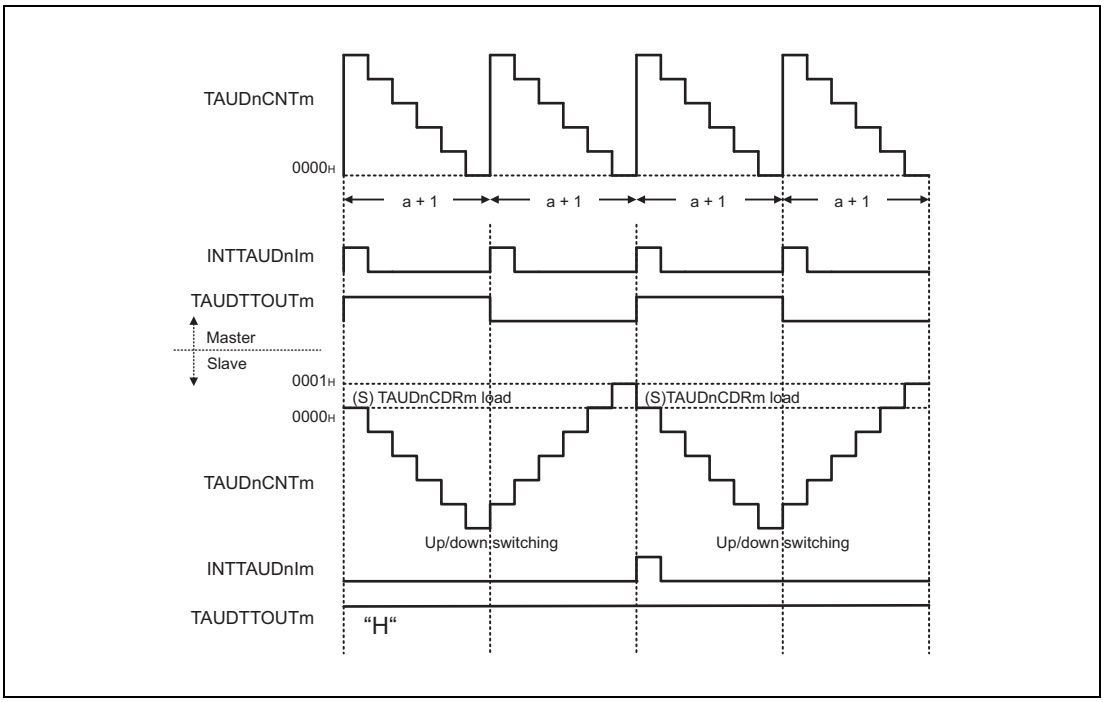



**Figure 25.119 TAUDnCDRm (Slave) = $0000_H$**

- If TAUDnCDRm (slave) = $0000_H$, INTTAUDnIm of the slave channel is not generated while counting up. TAUDTTOUTm remains high because there is no reset signal to be detected.

## 25.15.8 Triangle PWM Output Function with Dead Time

### 25.15.8.1 Overview

#### Summary

This function generates multiple triangle PWM outputs with a predefined dead time added by using a master and two or more slave channels. The resulting PWM signals with dead time are output via TAUDTTOUTm of the slave channels 2 and 3, enabling the pulse cycle (frequency) and the duty cycle of TAUDTTOUTm to be set using the master and slave channels.

Carrier cycles are generated on the master channel. The first pulse controls the down status of the slave counter and the second one controls the up status.

An interrupt on slave 2 causes TAUDTTOUTm of slave channels to be set/reset. Depending on the settings of TAUDnTDL.TAUDnTDLm, delay time is added to positive or negative logic side of the signal (i.e., whether TAUDTTOUTm is set/reset immediately or after dead time has elapsed). The duration of the dead time is specified by slave channel 3.

#### Prerequisites

- Three channels. For slave channels 2 and 3, select even channel CH (a) and odd channel CH (a + 1).
- The operating mode for the master channel should be set to interval timer mode. (See **Table 25.199, Contents of the TAUDnCMORm Register for the Master Channel of the Triangle PWM Output Function with Dead Time**)
- Slave channel 1 is not used for this function. This ensures that slave channel 2 is an even channel (a), and slave channel 3 is an odd channel (a + 1). Slave channel 1 can be used as a separate timer (independent function).
- The operating mode for slave channel 2 should be set to count-up/-down mode (See **Table 25.203, Contents of the TAUDnCMORm Register for Slave Channel 2 of the Triangle PWM Output Function with Dead Time**). Slave channel 2 should be an even channel.
- The operating mode for slave channel 3 should be set to one-count mode (See **Table 25.207, Contents of the TAUDnCMORm Register for Slave Channel 3 of the Triangle PWM Output Function with Dead Time**). Slave channel 3 should be an odd channel.
- The channel output mode for the master channel should be set to independent channel output mode 1. (See **Section 25.7, Channel Output Modes**)
- The output mode for slave channels 2 and 3 should be set to synchronous channel output mode 2 with dead time output. (See **Section 25.7, Channel Output Modes**)
- The following settings make a TAUDTTOUTm signal at high level during the down status of the carrier cycle:
  - If TAUDnCMORm.TAUDnMD0 (master) bit is set to 0, TAUDnTO.TAUDnTOm should be set to 1 while TAUDnTOE.TAUDnTOEm is set to 0 (recommended setting).
  - If TAUDnCMORm.TAUDnMD0 (master) bit is set to 1, TAUDnTO.TAUDnTOm should be set to 0 while TAUDnTOE.TAUDnTOEm is set to 0.

**NOTE**

The triangle PWM output function with dead time does not use slave channel 1. Slave channel 1 can be used as a separate timer (independent function).

### Functional description

The counter starts by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This makes TAUDnTE.TAUDnTEm = 1, enabling count operation. The current value of TAUDnCDRm is loaded into TAUDnCNTm and the counter starts to count down from the TAUDnCDRm value. If TAUDnCMORm.TAUDnMD0 bit of master channel is set to 1, an interrupt is generated and the master's TAUDTTOUTm signal is toggled.

- Master channel:
  When the counter of the master channel reaches $0000_H$, an INTTAUDnIm is generated and the TAUDTTOUTm signal is toggled. The TAUDnCDRm value is reloaded to continue counting down.
- Slave channel 2:
  If INTTAUDnIm is generated on the master channel, the counter of slave channel 2 is triggered.
  - If the slave counter is counting down, the counting direction changes.
  - If the slave counter is counting up, the TAUDnCDRm value is reloaded and the counter starts counting down.

  The counter continues to count down/up and waits for the next INTTAUDnIm of the master channel.
  When the counter value of slave channel 2 reaches $0001_H$, INTTAUDnIm is generated
- Slave channel 3:
  If INTTAUDnIm is generated on slave channel 2, the counter of slave channel 3 is triggered. The current value of TAUDnCDRm (slave 3) is loaded into TAUDnCNTm (slave 3) and the counter starts to count down from the TAUDnCDRm value.
  When the counter reaches $0000_H$, INTTAUDnIm occurs. The counter returns to $FFFF_H$ and waits for the next INTTAUDnIm of slave channel 2.

As described in **Table 25.198, Operation of TAUDTTOUTm upon Occurrence of an Interrupt on Slave Channel 2**, the set/reset timing (right after occurrence of an interrupt or after dead time has elapsed) depends on the TAUDnTDL.TAUDnTDLm setting of the corresponding channel.

The setting of TAUDnTOL.TAUDnTOLm also determines whether a high level signal (TAUDnTOL.TAUDnTOLm = 0) or a low level signal (TAUDnTOL.TAUDnTOLm = 1) is output from the corresponding channel.

The counter can be stopped by setting TAUDnTT.TAUDnTTm of master and slave channels to 1. This sets TAUDnTE.TAUDnTEm to 0. TAUDnCNTm and TAUDTTOUTm of master and slave channels stop but retain their values.

TAUDTTOUTm can be 100% output by setting the TAUDnCDRm value of slave channel 2 to $0000_H$.

**Conditions**

This function enables simultaneous rewrite. See **Section 25.6, Simultaneous Rewrite**.

TAUDnTOL.TAUDnTOLm and TAUDnTDL.TAUDnTDLm should be set before start of count operation. Slave channels 2 and 3 should have the opposite settings of TAUDnTDL.TAUDnTDLm.

**Table 25.198 Operation of TAUDTTOUTm upon Occurrence of an Interrupt on Slave Channel 2**

| TAUDnTDL. TAUDnTDLm | Count Direction of Slave Channel 2 upon Occurrence of Interrupt | TAUDTTOUTm Set/Reset Timing |
|---|---|---|
| 0 | Down | Set after elapse of dead time |
| | Up | Reset right after interrupt occurs |
| 1 | Down | Set right after interrupt occurs |
| | Up | Reset after elapse of dead time |

### 25.15.8.2 Equations

Pulse cycle = (TAUDnCDRm (master) + 1) × count clock cycle

$0000_H \leq$ TAUDnCDRm (master) $< FFFF_H$

Carrier cycle (down/up) = (TAUDnCDRm (master) + 1) × 2 × count clock cycle

PWM signal width (normal phase) = [(TAUDnCDRm (master) + 1 – TAUDnCDRm (slave 2) )× 2 – (TAUDnCDRm (slave 3) + 1)] × count clock cycle

PWM signal width (reverse phase) = [(TAUDnCDRm (master) + 1 – TAUDnCDRm (slave 2)) × 2 + (TAUDnCDRm (slave 3) + 1)] × count clock cycle

### 25.15.8.3 Block Diagram and General Timing Diagram

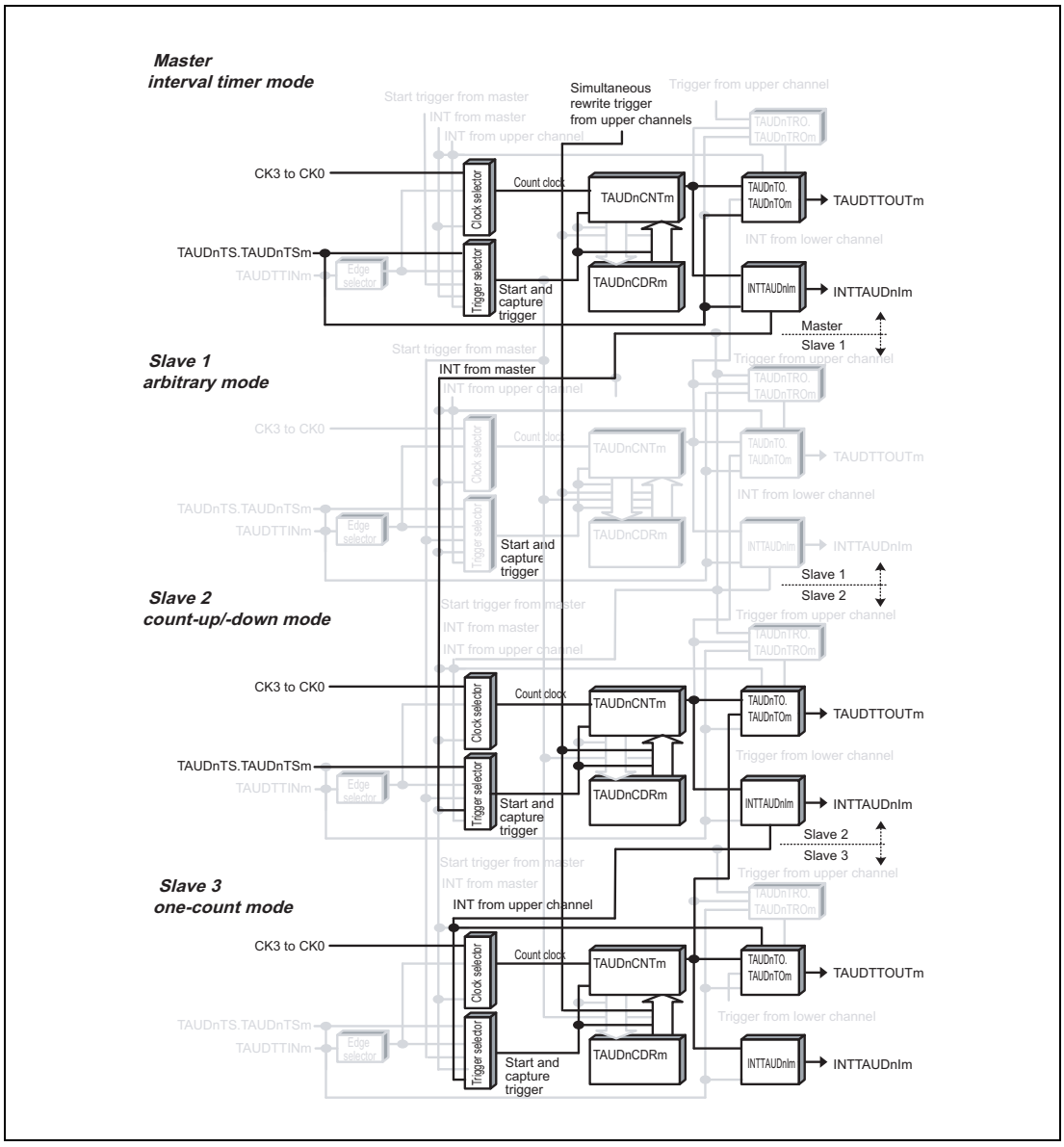



**Figure 25.120 Block Diagram of Triangle PWM Output Function with Dead Time**

The following settings apply to the general timing diagram.

- Master channel:
  - INTTAUDnIm is generated at the beginning of operation. (TAUDnCMORm.TAUDnMD0 = 1)
- Slave channel 2:
  - INTTAUDnIm not generated at the beginning of operation. (TAUDnCMORm.TAUDnMD0 = 0)
  - TAUDnTDL.TAUDnTDLm = 0
  - Positive logic (TAUDnTOL.TAUDnTOLm = 0)

- Slave channel 3:
  - Enables start trigger detection during counting (TAUDnCMORm.TAUDnMD0 = 1)
  - TAUDnTDL.TAUDnTDLm = 1
  - Positive logic (TAUDnTOL.TAUDnTOLm = 0)

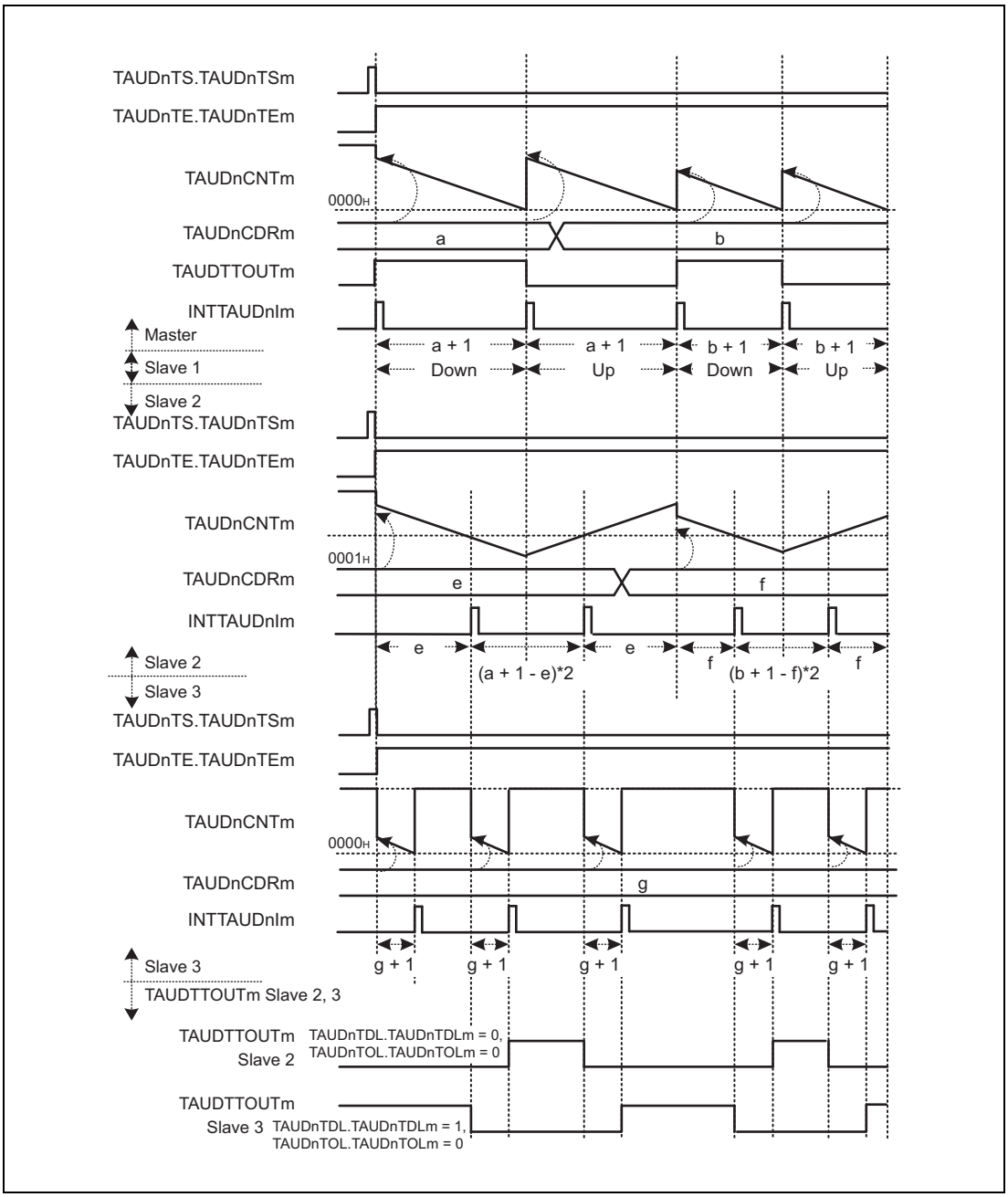



**Figure 25.121 General Timing Diagram of Triangle PWM Output Function with Dead Time**

### 25.15.8.4 Register Settings for the Master Channel

#### (1) TAUDnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.199 Contents of the TAUDnCMORm Register for the Master Channel of the Triangle PWM Output Function with Dead Time**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 1: Master channel |
| 10 to 8 | TAUDnSTS[2:0] | 000: Trigger the counter using software. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0000: Interval timer mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm is not generated and TAUDTTOUTm is not toggled at the beginning of operation.<br>1: INTTAUDnIm is generated and TAUDTTOUTm is toggled at the beginning of operation. |

#### (2) TAUDnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.200 Contents of the TAUDnCMURm Register for the Master Channel of the Triangle PWM Output Function with Dead Time**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for the master channel

Table 25.201 Control Bit Settings in Independent Channel Output Mode 1

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 0: Independent channel output |
| TAUDnTOC.TAUDnTOCm | 0: Operating mode 1 (Toggle mode if TAUDnTOM.TAUDnTOMm = 0) |
| TAUDnTOL.TAUDnTOLm | 0: The setting is disabled in toggle mode (the value after reset). |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set this bit to 0 |
| TAUDnTRC.TAUDnTRCm | 0: Disables the operation as a real-time output trigger channel |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

### (4) Simultaneous rewrite for the master channel

Both the master and slave channels should have the same simultaneous rewrite settings.

Table 25.202 Simultaneous Rewrite Setting for the Master Channel of Triangle PWM Output Function with Dead Time

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Selects a master channel for simultaneous rewrite triggers.<br>1: Selects an upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 1: A simultaneous rewrite trigger signal is generated when master channel starts to count and the corresponding slave channel is at the peak of a triangular wave cycle. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

**NOTE**

If TAUDnRDS.TAUDnRDSm = 1, it is necessary for an upper channel higher than the master channel to generate a simultaneous rewrite trigger signal.

### 25.15.8.5 Register Settings for Slave Channel 2

#### (1) TAUDnCMORm for slave channel 2

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.203 Contents of the TAUDnCMORm Register for Slave Channel 2 of the Triangle PWM Output Function with Dead Time**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 111: Up/down output trigger signal of master channel |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 1001: Count-up/-down mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm not generated at the beginning of operation. |

#### (2) TAUDnCMURm for slave channel 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.204 Contents of the TAUDnCMURm Register for Slave Channel 2 of the Triangle PWM Output Function with Dead Time**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for slave channel 2

Table 25.205 Control Bit Settings in Synchronous Channel Output Mode 2 with Dead Time Output

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 1: Synchronous channel operation |
| TAUDnTOC.TAUDnTOCm | 1: Operating mode 2 |
| TAUDnTOL.TAUDnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUDnTDE.TAUDnTDEm | 1: Enables dead time operation. |
| TAUDnTDM.TAUDnTDMm | 0: Adds dead time if an interrupt is detected on an even upper channel and the conditions set by TAUDnTDL.TAUDnTDLm are satisfied. |
| TAUDnTDL.TAUDnTDLm | 0: Adds dead time to normal phase.<br>1: Adds dead time to reverse phase. |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TREm = 0), set this bit to 0 |
| TAUDnTRC.TAUDnTRCm | 0: Disables the operation as a real-time output trigger channel |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

**CAUTION**

Set TAUDnTDL.TAUDnTDLm exclusively from odd channels.

### (4) Simultaneous rewrite for slave channel 2

Both the master and slave channels should have the same simultaneous rewrite settings.

Table 25.206 Simultaneous Rewrite Settings for Slave Channel 2 of Triangle PWM Output Function with Dead Time

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Selects a master channel for simultaneous rewrite triggers.<br>1: Selects an upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 1: A simultaneous rewrite trigger signal is generated when master channel starts to count and the corresponding slave channel is at the peak of a triangular wave cycle. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

### 25.15.8.6 Register Settings for Slave Channel 3

#### (1) TAUDnCMORm for slave channel 3

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.207 Contents of the TAUDnCMORm Register for Slave Channel 3 of the Triangle PWM Output Function with Dead Time**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 110: Dead time output signal of the TAUDTTOUTm generation unit |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0100: One-count mode |
| 0 | TAUDnMD0 | 1: Enables start trigger detection while counting. |

#### (2) TAUDnCMURm for slave channel 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.208 Contents of the TAUDnCMURm Register for Slave Channel 3 of the Triangle PWM Output Function with Dead Time**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for slave channel 3

Table 25.209 Control Bit Settings in Synchronous Channel Output Mode 2 with Dead Time Output

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 1: Synchronous channel operation |
| TAUDnTOC.TAUDnTOCm | 1: Operating mode 2 |
| TAUDnTOL.TAUDnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUDnTDE.TAUDnTDEm | 1: Enables dead time operation. |
| TAUDnTDM.TAUDnTDMm | 0: Adds dead time if an interrupt is detected on an even upper channel and the conditions set by TAUDnTDL.TAUDnTDLm are satisfied. |
| TAUDnTDL.TAUDnTDLm | 0: Adds dead time to normal phase.<br>1: Adds dead time to reverse phase. |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set this bit to 0 |
| TAUDnTRC.TAUDnTRCm | 0: Disables the operation as a real-time output trigger channel |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

**CAUTION**

Set TAUDnTDL.TAUDnTDLm exclusively from even channels.

### (4) Simultaneous rewrite for slave channel 3

Both the master and slave channels should have the same simultaneous rewrite settings.

Table 25.210 Simultaneous Rewrite Settings for Slave Channel 3 of Triangle PWM Output Function with Dead Time

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Selects a master channel for simultaneous rewrite triggers.<br>1: Selects an upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 1: A simultaneous rewrite trigger signal is generated when master channel starts to count and the corresponding slave channel is at the peak of a triangular wave cycle. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

### 25.15.8.7 Operating Procedure for Triangle PWM Output Function with Dead Time

Table 25.211 Operating Procedure for Triangle PWM Output Function with Dead Time

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Master channel: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.8.4, Register Settings for the Master Channel**.<br><br>Slave channel 2: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.8.5, Register Settings for Slave Channel 2**.<br><br>Slave channel 3: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.8.6, Register Settings for Slave Channel 3**.<br><br>Set the value of TAUDnCDRm register of every channel. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm of master and slave channels to 1 simultaneously.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm (master and slave channels) is set to 1 and the counters of master and slave channels start.<br>INTTAUDnIm (master) is generated on the master channel if TAUDnCMORm.TAUDnMD0 is set to 1. |
| During Operation | TAUDnCDRm can be changed at any time.<br>TAUDnCNTm and TAUDnRSF.TAUDnRSFm can be read at any time.<br><br>TAUDnRDT.TAUDnRDTm can be changed during operation. | TAUDnCDRm value of master channel and slave channel 2 is loaded into TAUDnCNTm to perform counting down. When the counter of master channel reaches $0000_H$:<br>• INTTAUDnIm (master) is generated.<br>• TAUDnCDRm value is reloaded into TAUDnCNTm (master) to continue count operation.<br>• TAUDnCDRm value is reloaded into TAUDnCNTm (slave 2) or counting is started in opposite direction.<br>When TAUDnCNTm of slave channel 2 reaches $0001_H$:<br>• INTTAUDnIm (slave 2) is generated.<br>• TAUDnCDRm value of slave channel 3 is loaded into TAUDnCNTm perform counting down.<br>When TAUDnCNTm of slave channel 3 reaches $0000_H$:<br>• INTTAUDnIm is generated. |
| Stop Operation | Set TAUDnTT.TAUDnTTm of master and slave channels to 1 simultaneously.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm and TAUDTTOUTm stop and retain their current values. |

Restart Operation

### 25.15.8.8 Specific Timing Diagrams

#### (1) Duty cycle = 0%

The following settings apply to the general timing diagram in **Figure 25.122**.

- Slave channel 2:
  - Positive logic (TAUDnTDL.TAUDnTDLm = 0)
- Slave channel 3:
  - Negative logic (TAUDnTDL.TAUDnTDLm = 1)

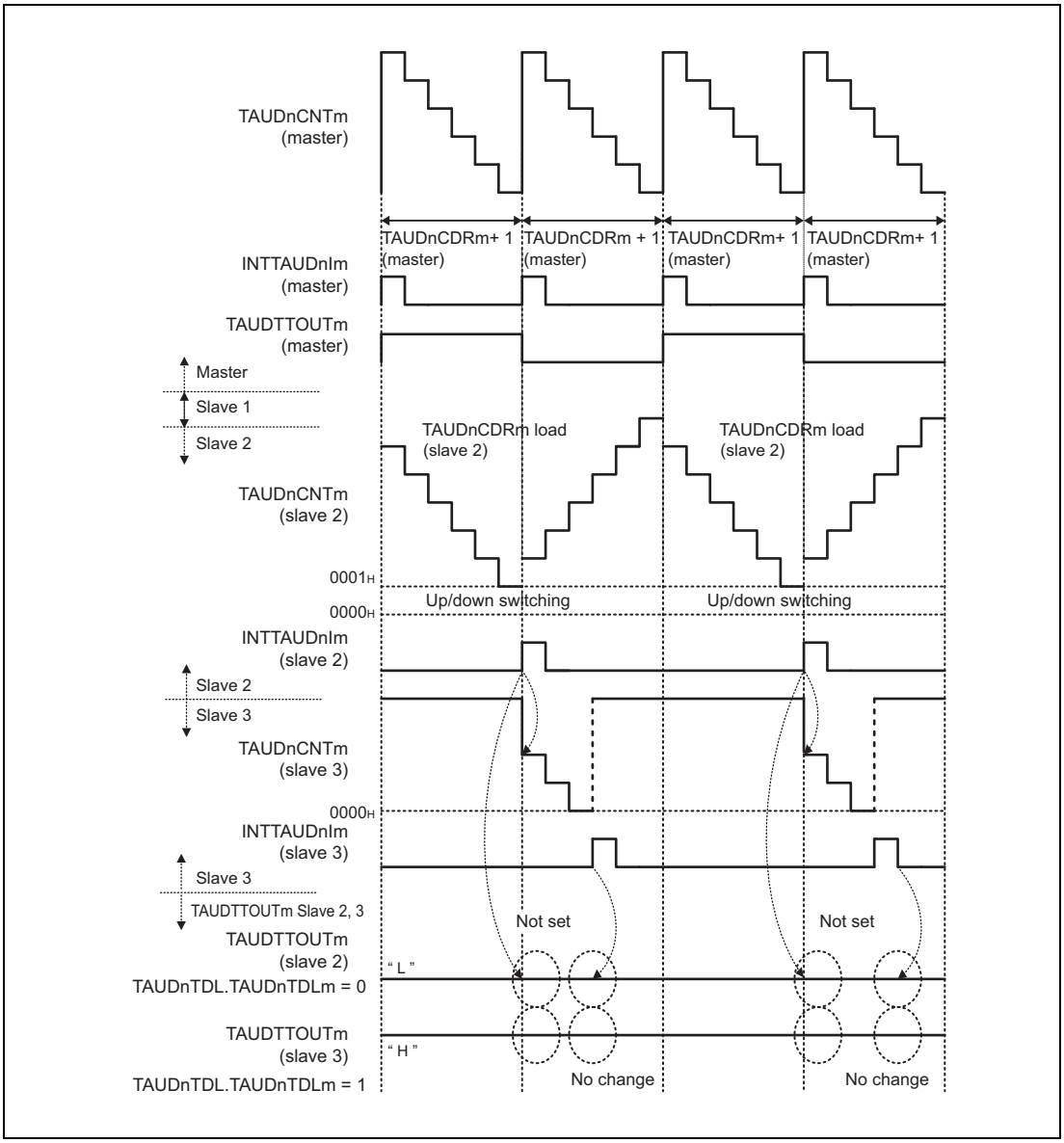



**Figure 25.122 TAUDnCDRm (Slave 2) ≥ TAUDnCDRm (Master) + 1**

- If TAUDnCDRm (slave 2) is greater than TAUDnCDRm (master), the counter of slave channel does not reach $0000_H$ while counting down. Therefore, TAUDTTOUTm signal is not set/reset and remains in the initial state. This signal becomes a reset signal because an interrupt occurs on slave channel 2 during count-up operation.

### (2) Duty cycle = 100%

The following settings apply to the general timing diagram in **Figure 25.123**.

- Slave channel 2:
  - Positive logic (TAUDnTDL.TAUDnTDLm = 0)
- Slave channel 3:
  - Negative logic (TAUDnTDL.TAUDnTDLm = 1)

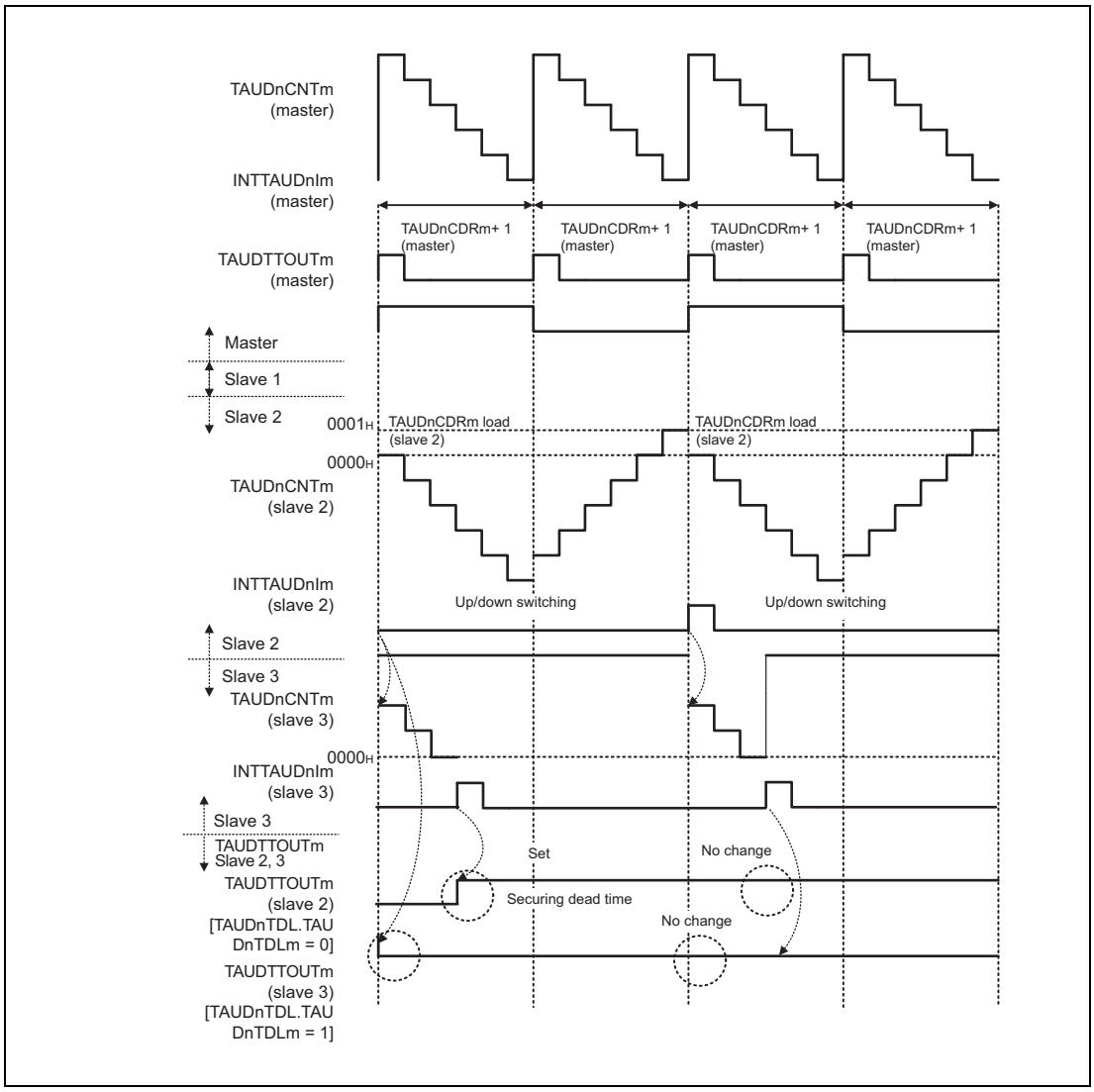



**Figure 25.123 TAUDnCDRm (Slave) = $0000_H$**

- If TAUDnCDRm (slave 2) = $0000_H$, the slave channel counter does not reach $0001_H$ while counting up. Therefore, no INTTAUDnIm is generated during count-up operation.
  - The set conditions for a channel with TAUDnTDL.TAUDnTDLm = 0 are met after elapse of dead time. TAUDTTOUTm is left in a newly set state even if a set/reset is made because no reset conditions are satisfied on such a channel.
  - Slave channel 3 in the above diagram is set when the counter starts. However, TAUDTTOUTm is left in an initial state on the slave channel with TAUDnTDL.TAUDnTDLm = 1 because no reset conditions are satisfied on that channel.

## 25.15.9 A/D Conversion Trigger Output Function Type 2

### 25.15.9.1 Overview

#### Summary

This function is identical to **Section 25.15.7, Triangle PWM Output Function**, except that TAUDTTOUTm is not output.

This function is enabled by setting channel output mode for the slave to independent channel output mode controlled by software.

### 25.15.9.2 Block Diagram and General Timing Diagram

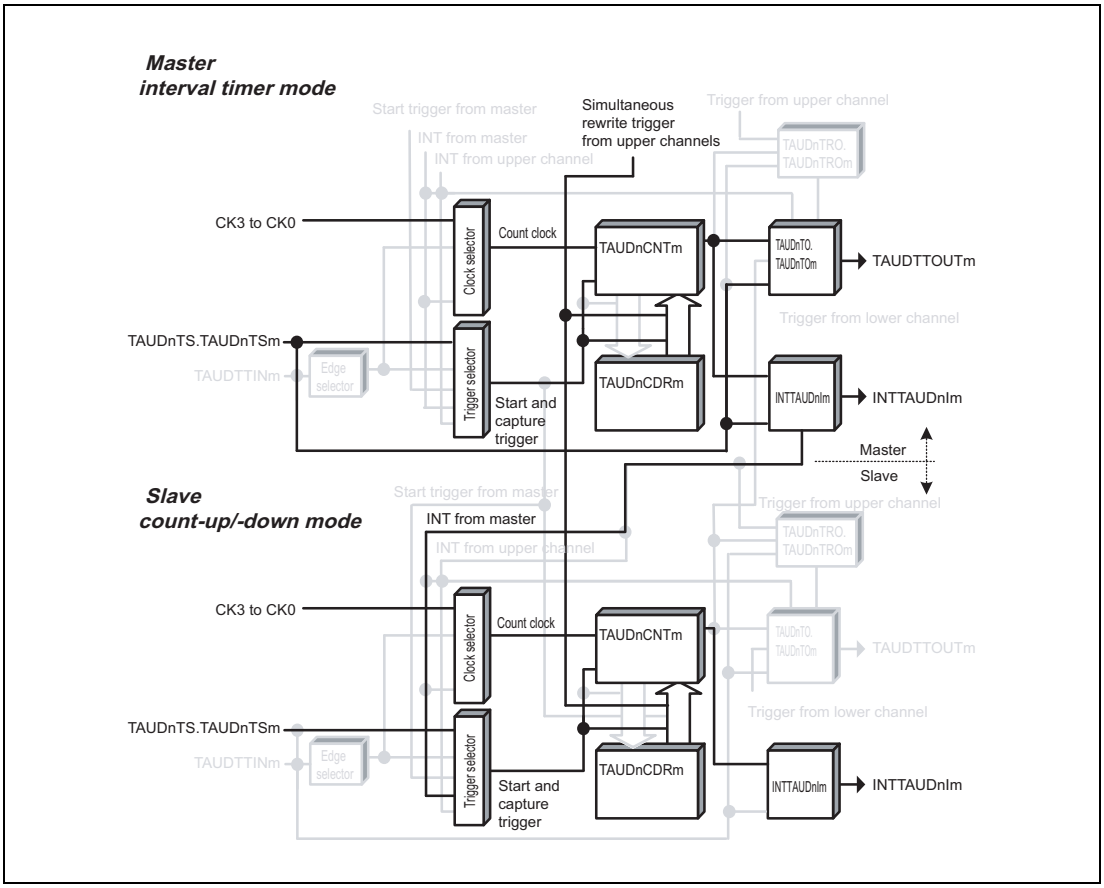



**Figure 25.124 Block Diagram of A/D Conversion Trigger Output Function Type 2**

The following settings apply to the general timing diagram.

- Master channel
  - INTTAUDnIm is generated at the beginning of operation.
    (TAUDnCMORm.TAUDnMD0 = 1)

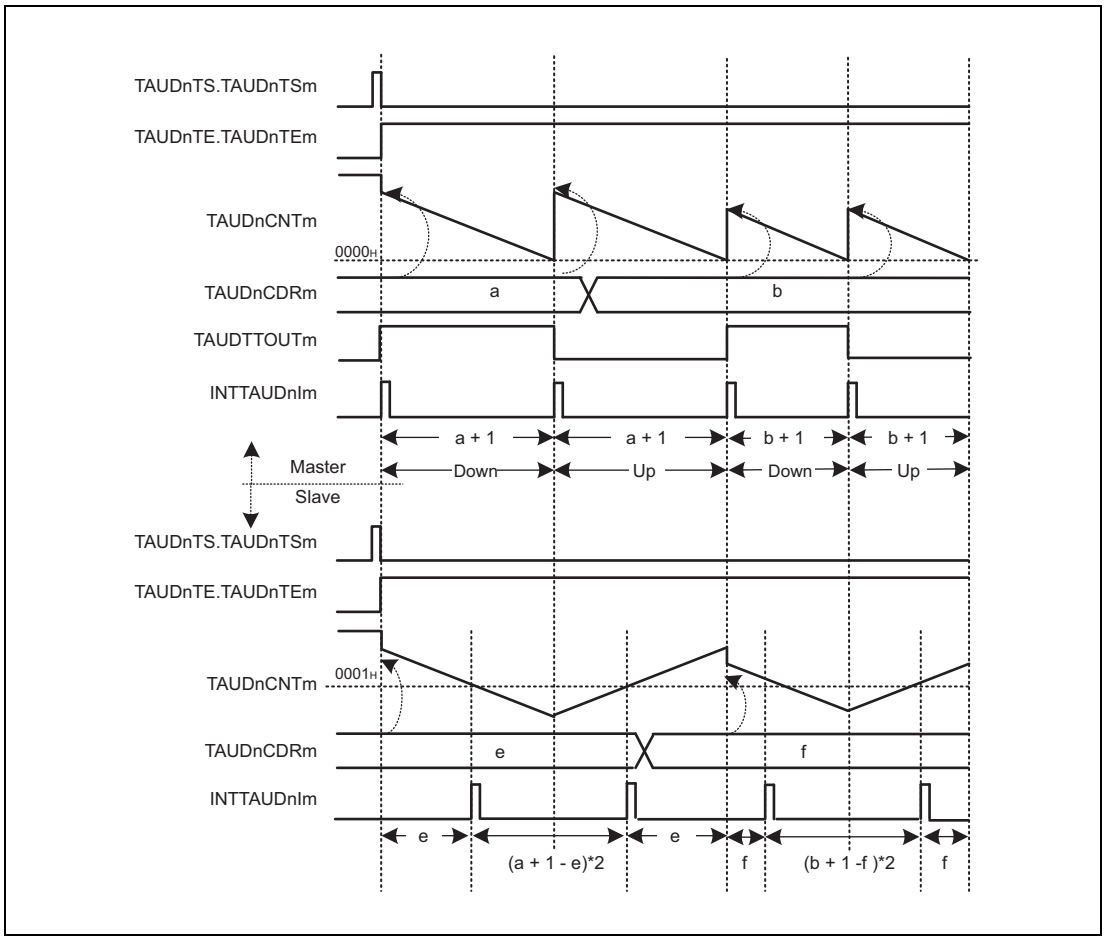



**Figure 25.125 General Timing Diagram of A/D Conversion Trigger Output Function Type 2**

## 25.15.10 Interrupt Request Signals Culling Function

### 25.15.10.1 Overview

#### Summary

This function divides the number of interrupts of the master channel by a specified value using a slave channel.

The interrupt request signals culling function is a sub function of the following functions:

- PWM Output Function (See **Section 25.15.1, PWM Output Function**)
- Triangle PWM Output Function (See **Section 25.15.7, Triangle PWM Output Function**
- Triangle PWM Output Function with Dead Time
  (See **Section 25.15.8, Triangle PWM Output Function with Dead Time**)

#### Prerequisites

- Two channels
- The operation mode of the master channel must be set to interval timer mode. (See **Table 25.212, Contents of the TAUDnCMORm Register for the Master Channel of the Interrupt Request Signals Culling Function**)
- The operation mode of the slave channel must be set to Event Count Mode. (See **Table 25.215, Contents of the TAUDnCMORm Register for the Slave Channel of the Interrupt Request Signals Culling Function**)
- This function does not use TAUDTTOUTm.

#### Functional description

The counters (master and slave) are enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1 for both channels. This in turn sets TAUDnTE.TAUDnTEm, enabling count operation. The current value of the data register of the master channel and slave channel (TAUDnCDRm) are written to the counter (TAUDnCNTm).

- Master channel:
  When the counter of the master channel reaches $0000_H$, INTTAUDnIm is generated and TAUDnCDRm value is reloaded to TAUDnCNTm.
- Slave channel:
  Every time the master channel generates an INTTAUDnIm, the counter of the slave channel decrements by one. When the counter reaches $0000_H$, it awaits the next interrupt from the master channel. This causes TAUDnCNTm (slave) to reload the value of TAUDnCDRm, and an INTTAUDnIm is generated.

Forced restart is not possible for this function. The counter can be stopped by setting TAUDnTT.TAUDnTTm to 1 for the master and slave channels, which in turn sets TAUDnTE.TAUDnTEm to 0. TAUDnCNTm of master and slave channels stops but retains its value.

#### Conditions

This function enables simultaneous rewrite. See **Section 25.6, Simultaneous Rewrite**.

### 25.15.10.2 Equations

Interrupt division operator = TAUDnCDRm (slave channel)

– One INTTAUDnIm is generated for the INTTAUDnIm count of the master channel defined by TAUDnCDRm (slave channel) + 1.

### 25.15.10.3 Block Diagram and General Timing Diagram

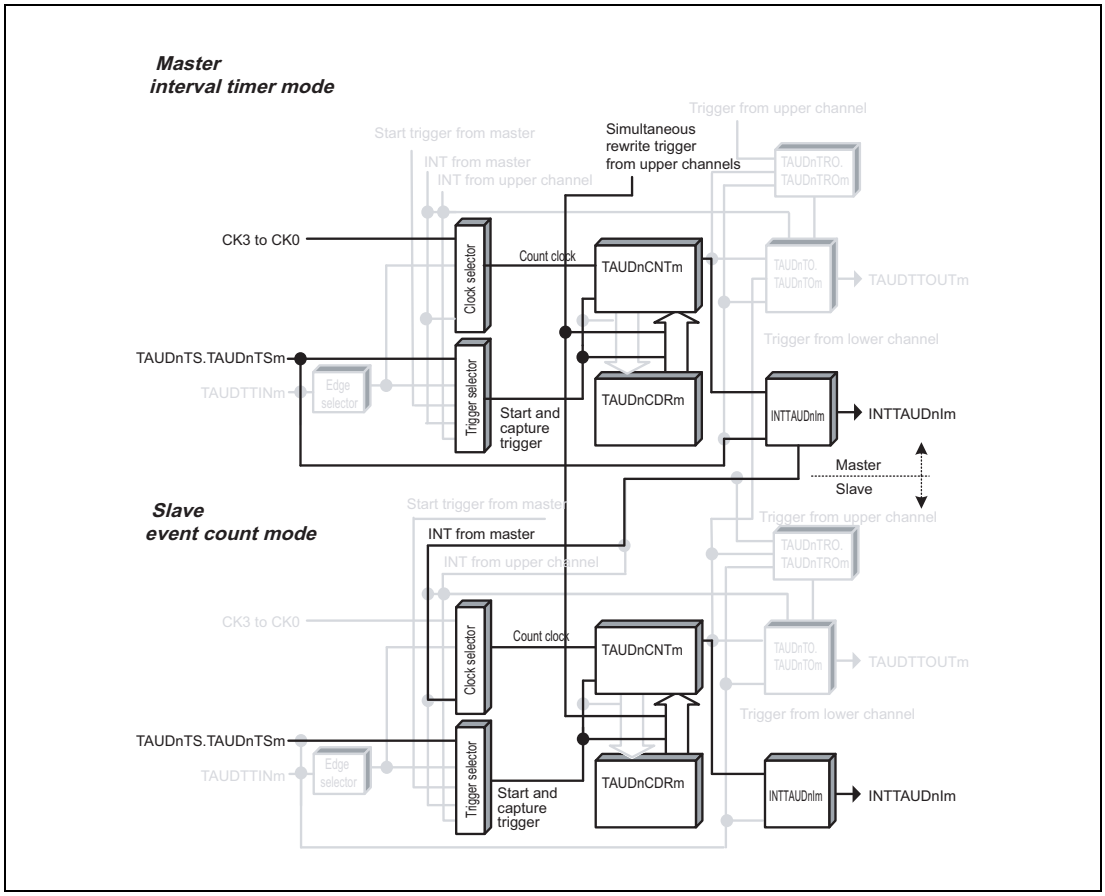



**Figure 25.126 Block Diagram of Interrupt Request Signals Culling Function**

The following settings apply to the general timing diagram.

Master channel:

- INTTAUDnIm is generated at the beginning of operation. (TAUDnCMORm.TAUDnMD0 = 1)

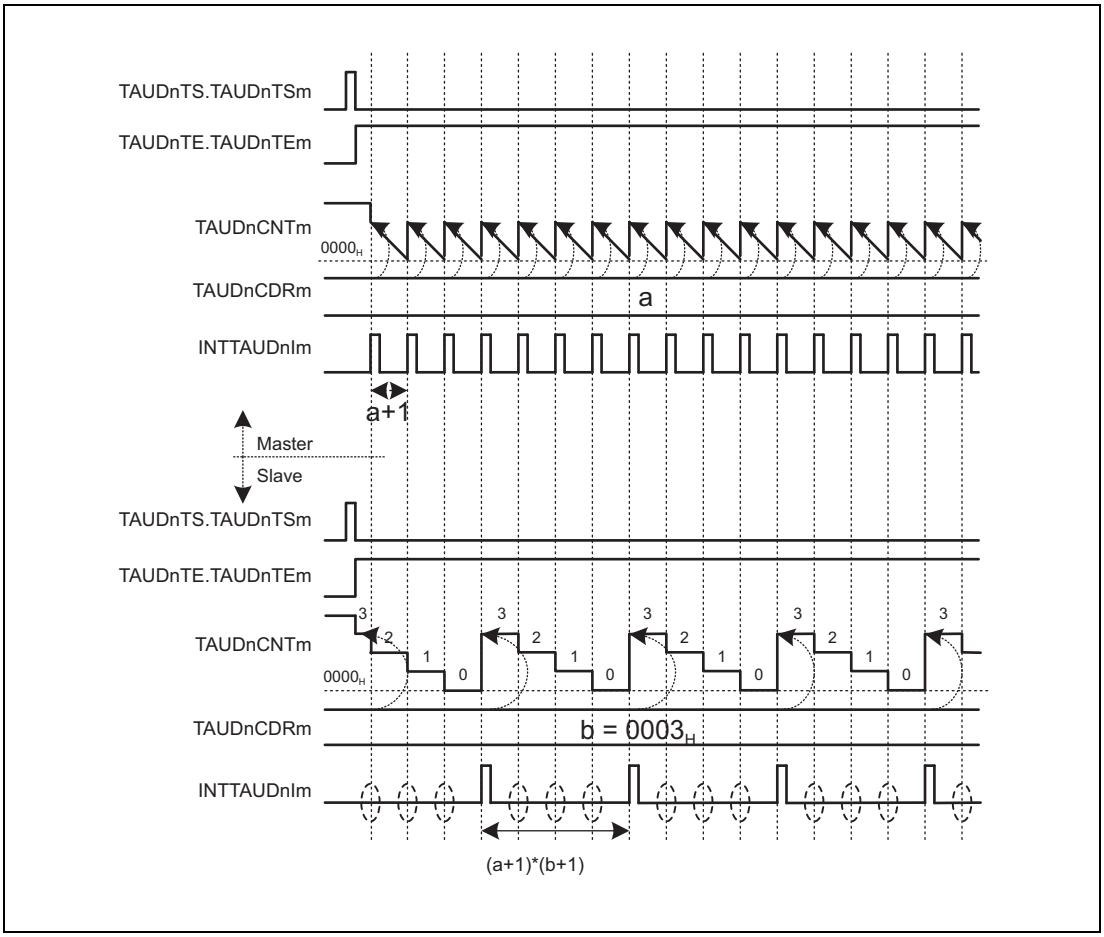



**Figure 25.127 General Timing Diagram of Interrupt Request Signals Culling Function**

### 25.15.10.4 Register Settings for the Master Channel

#### (1) TAUDnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.212 Contents of the TAUDnCMORm Register for the Master Channel of the Interrupt Request Signals Culling Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 1: Master channel |
| 10 to 8 | TAUDnSTS[2:0] | 000: Trigger the counter using software. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0000: Interval timer mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm not generated at the beginning of operation.<br>1: INTTAUDnIm generated at the beginning of operation. |

#### (2) TAUDnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.213 Contents of the TAUDnCMURm Register for the Master Channel of the Interrupt Request Signals Culling Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for the master channel

TAUDnTOE.TAUDnTOEm is set to 0 because the channel output mode is not used with this function.

### (4) Simultaneous rewrite for the master channel

Both the master and slave channels should have the same simultaneous rewrite settings.

**Table 25.214 Simultaneous Rewrite Settings for the Master Channel of Interrupt Request Signals Culling Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Selects a master channel for simultaneous rewrite triggers.<br>1: Selects an upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 0: Generates a simultaneous rewrite trigger signal when the master channel starts to count.<br>1: Simultaneous rewrite trigger signal is generated when master channel counter is started and the corresponding slave channel is at the peak of triangular wave. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

### 25.15.10.5 Register Settings for the Slave Channel

#### (1) TAUDnCMORm for the slave channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.215 Contents of the TAUDnCMORm Register for the Slave Channel of the Interrupt Request Signals Culling Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 11: INTTAUDnIm of the master channel is used as the count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 000: Trigger the counter using software. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0011: Event count mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm not generated at the beginning of operation. |

#### (2) TAUDnCMURm for the slave channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.216 Contents of the TAUDnCMURm Register for the Slave Channel of the Interrupt Request Signals Culling Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

#### (3) Channel output mode for the slave channel

TAUDnTOE.TAUDnTOEm is set to 0 because the channel output mode is not used with this function.

#### (4) Simultaneous rewrite for the slave channel

Both the master and slave channels should have the same simultaneous rewrite settings.

**Table 25.217 Simultaneous Rewrite Settings for the Slave Channel of Interrupt Request Signals Culling Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Selects a master channel for simultaneous rewrite triggers.<br>1: Selects an upper channel outside the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 0: Generates a simultaneous rewrite trigger signal when the master channel starts to count.<br>1: Simultaneous rewrite trigger signal is generated when master channel counter is started and the corresponding slave channel is at the peak of triangular wave. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

### 25.15.10.6 Operating Procedure for Interrupt Request Signals Culling Function

**Table 25.218 Operating Procedure for Interrupt Request Signals Culling Function**

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Master channel: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.10.4, Register Settings for the Master Channel**.<br><br>Slave channel: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.15.10.5, Register Settings for the Slave Channel**.<br><br>Set the value of TAUDnCDRm register of every channel. | Channel operation is stopped. |
| Start Operation | Set TAUDnTS.TAUDnTSm of master and slave channels to 1 simultaneously.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm (master and slave channels) is set to 1 and the counters of master and slave channels start.<br>INTTAUDnIm is generated on the master channel. |
| During Operation | TAUDnCDRm can be changed at any time.<br>TAUDnCNTm and TAUDnRSF.TAUDnRSFm can be read at any time.<br><br>TAUDnRDT.TAUDnRDTm can be changed during operation. | TAUDnCNTm of master channel loads TAUDnCDRm value and counts down. When the counter reaches $0000_H$:<br>• INTTAUDnIm (master) is generated.<br>• TAUDnCNTm (master) loads TAUDnCDRm value and continues count operation.<br>• TAUDnCNTm of slave channels counts down each time INTTAUDnIm of master channel is detected.<br>When TAUDnCNTm of the slave = $0000_H$:<br>• INTTAUDnIm (slave) is generated.<br>• The TAUDnCDRm value is loaded in TAUDnCNTm (slave) and count operation continues. |
| Stop Operation | Set TAUDnTT.TAUDnTTm of master and slave channels to 1 simultaneously.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm stops and retains its current value. |

Restart Operation

### 25.15.10.7 Specific Timing Diagrams

#### (1) Interrupt count (master) = interrupt count (slave)

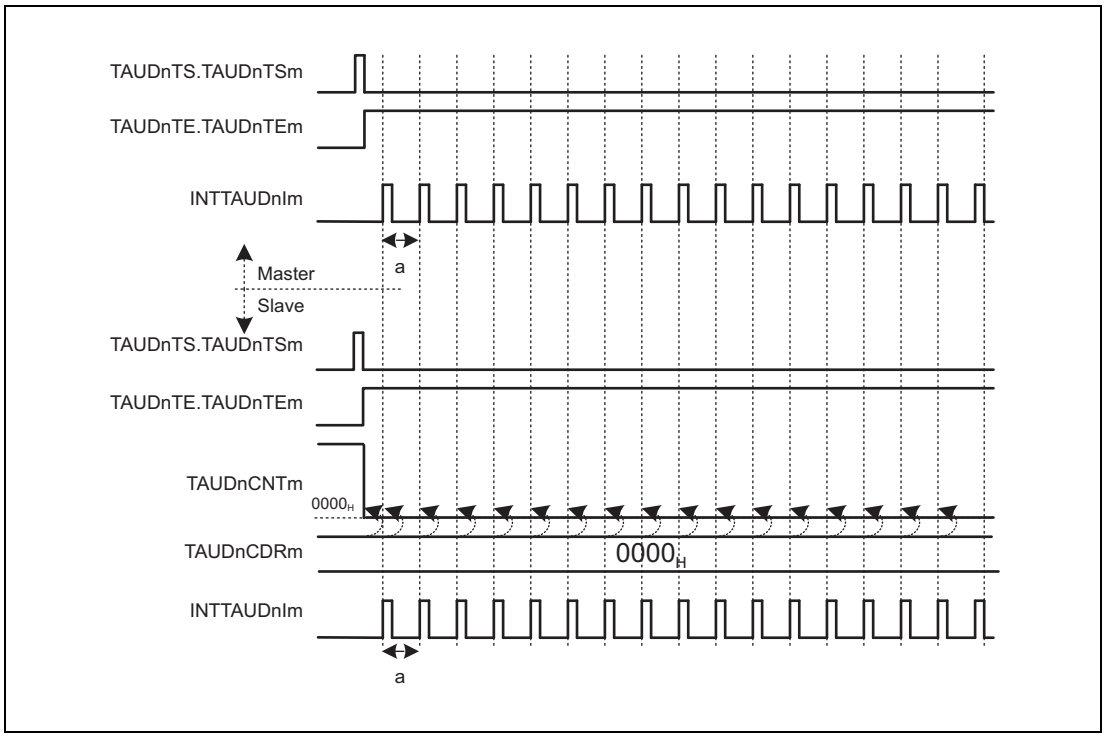



Figure 25.128 TAUDnCDRm (Slave) = $0000_H$

- If TAUDnCDRm = $0000_H$, the TAUDnCDRm value of the slave channel is loaded into TAUDnCNTm each time INTTAUDnIm of master channel is detected. In other words, TAUDnCNTm is always $0000_H$.
- Therefore, an interrupt occurs on the master channel and simultaneously an interrupt occurs on slave channels.

# 25.16 Synchronous Non-Complementary and Complementary Modulation Output Functions

This section describes functions that generate 6-phase PWM output or triangle PWM output using a master channel and seven slave channels.

## 25.16.1 Non-Complementary Modulation Output Function Type 1

### 25.16.1.1 Overview

#### Summary

This function outputs a PWM signal, a high-level signal, or a low-level signal from TAUDTTOUTm depending on the values of the real-time output bits (TAUDnTRO.TAUDnTROm) and the modulation output enable bits (TAUDnTME.TAUDnTMEm) of a pair of slave channels. Three pairs of channels are typically used.

#### Prerequisites

- One master channel and seven slave channels
- The operation mode of the master channel must be set to interval timer mode (See **Table 25.220, Contents of the TAUDnCMORm Register for the Master Channel of Non-Complementary Modulation Output Function Type 1**).
- The operating mode for slave channels 1 to 7 should be set to one-count mode (See **Table 25.223, Contents of the TAUDnCMORm Register for Slave Channel 1 of Non-Complementary Modulation Output Function Type 1**, and **Table 25.226, Contents of the TAUDnCMORm Register for Slave Channels 2 to 7 of Non-Complementary Modulation Output Function Type 1**).
- TAUDTTOUTm is not used with the master channel of this function.
- TAUDTTOUTm of slave channel 1 is not used with this function, but TAUDnTRC.TAUDnTRCm should be set to 1 (See **Section 25.7, Channel Output Modes**).
- The channel output mode for slave channels 2 to 7 should be set to synchronous channel output mode 1 with non-complementary modulation output (See **Section 25.7, Channel Output Modes**).
- TAUDnCDRm of slave channel 1 should be set to $0000_H$.

#### Functional description

The master/slave channel counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This sets TAUDnTE.TAUDnTEm = 1, enabling count operation. The value of data register (TAUDnCDRm) is loaded into the counter (TAUDnCNTm) and the counter starts to count down. When the counter reaches $0000_H$, INTTAUDnIm is generated.

- Slave channel 1:
  Slave channel 1 is set as a channel that triggers real-time output (TAUDnTRC.TAUDnTRCm = 1). If an interrupt occurs on slave channel 1 (TAUDnCDRm is fixed to $0000_H$), the value of real-time output bit (TAUDnTRO.TAUDnTROm) of the channel that monitors the interrupt on slave channel 1 is reflected to the TAUDTTOUTm output. After that, the counter returns to $FFFF_H$ and waits for the next interrupt of master channel.

- Slave channel 2:
  Slave channel 2 generates a PWM output. The master channel specifies a PWM output cycle and slave channel 2 specifies a duty cycle. After generating an interrupt, the counter returns to $FFFF_H$ and awaits the next interrupt from the master channel.

Slave channels 3 to 7 operate like slave channel 2.

As described in **Table 25.219, TAUDTTOUTm Output of Slave Channels for Non-Complementary Modulation Output Function Type 1 (TAUDnTOL.TAUDnTOLm = 0)**, a signal output from TAUDTTOUTm depends on the value of the real-time output bit (TAUDnTRO.TAUDnTROm) and modulation output bit (TAUDnTME.TAUDnTMEm) of slave channel.

This function cannot use a forced restart. The counter can be stopped by setting TAUDnTT.TAUDnTTm of master and slave channels to 1. This sets TAUDnTE.TAUDnTEm to 0. TAUDnCNTm and TAUDTTOUTm of master and slave channels stop but retain their values. The counters can be restarted by setting TAUDnTS.TAUDnTSm to 1.

### Conditions

- If TAUDnTME.TAUDnTMEm = 0 on slave channels 2 to 7 (TAUDnTOL.TAUDnTOLm = 0):
  - If the channel's TAUDnTRO.TAUDnTROm is set to 1, TAUDTTOUTm outputs a high-level signal.
  - If the channel's TAUDnTRO.TAUDnTROm is set to 0, TAUDTTOUTm outputs a low-level signal.
- If TAUDnTME.TAUDnTMEm = 1 on slave channels 2 to 7 (TAUDnTOL.TAUDnTOLm = 0):
  - If the channel's TAUDnTRO.TAUDnTROm is set to 1, TAUDTTOUTm outputs PWM (positive logic) corresponding to the channel.
  - If the channel's TAUDnTRO.TAUDnTROm is set to 0, TAUDTTOUTm outputs a low-level signal.
- If TAUDnTOL.TAUDnTOLm is set to 1, high-level and low-level signals output from TAUDTTOUTm are inverted. The PWM signal is negative logic. Only the initial setting of TAUDnTOL.TAUDnTOLm is permitted (cannot be changed during operation).

**Table 25.219 TAUDTTOUTm Output of Slave Channels for Non-Complementary Modulation Output Function Type 1 (TAUDnTOL.TAUDnTOLm = 0)**

| TAUDnTME.TAUDnTMEm | TAUDnTRO.TAUDnTROm | TAUDTTOUTm Output |
|---|---|---|
| 0 | 0 | Low level |
| | 1 | High level |
| 1 | 0 | Low level |
| | 1 | PWM (positive logic) |

- This function enables simultaneous rewrite. See **Section 25.6, Simultaneous Rewrite**.
- TAUDnCDRm value of slave channel 1 should be set to $0000_H$ so that a real-time output is triggered at the same time with PWM generation on slave channels 2 to 7.
- If TAUDnTOL.TAUDnTOLm is set to 0 on slave channels 2 to 7, TAUDnTO.TAUDnTOm is set to 0 (low) before TAUDnTE.TAUDnTEm is set to 0.
- If TAUDnTOL.TAUDnTOLm is set to 1 on slave channels 2 to 7, TAUDnTO.TAUDnTOm is set to 1 (high) before TAUDnTE.TAUDnTEm is set to 0.

#### 25.16.1.2 Equations

Slave channels 2 to 7:

Pulse period = [TAUDnCDRm (master) + 1] × count clock cycle

Duty time = [TAUDnCDRm (slave)] × count clock cycle

### 25.16.1.3 Block Diagram and General Timing Diagram

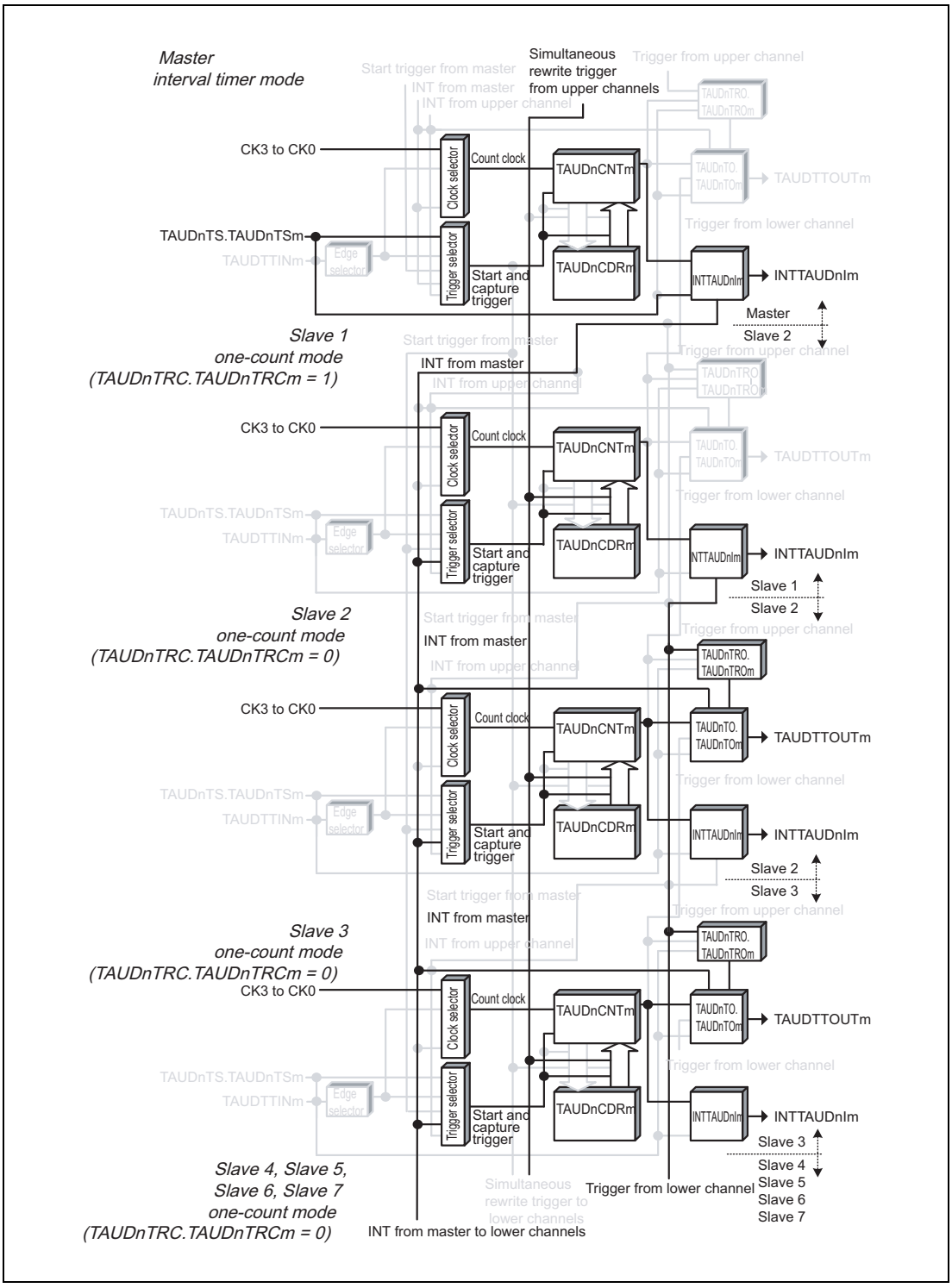



**Figure 25.129 Block Diagram of Non-Complementary Modulation Output Function Type 1**

The following settings apply to the general timing diagram.

- Slave channels 2 to 7: Positive logic (TAUDnTOL.TAUDnTOLm = 0)

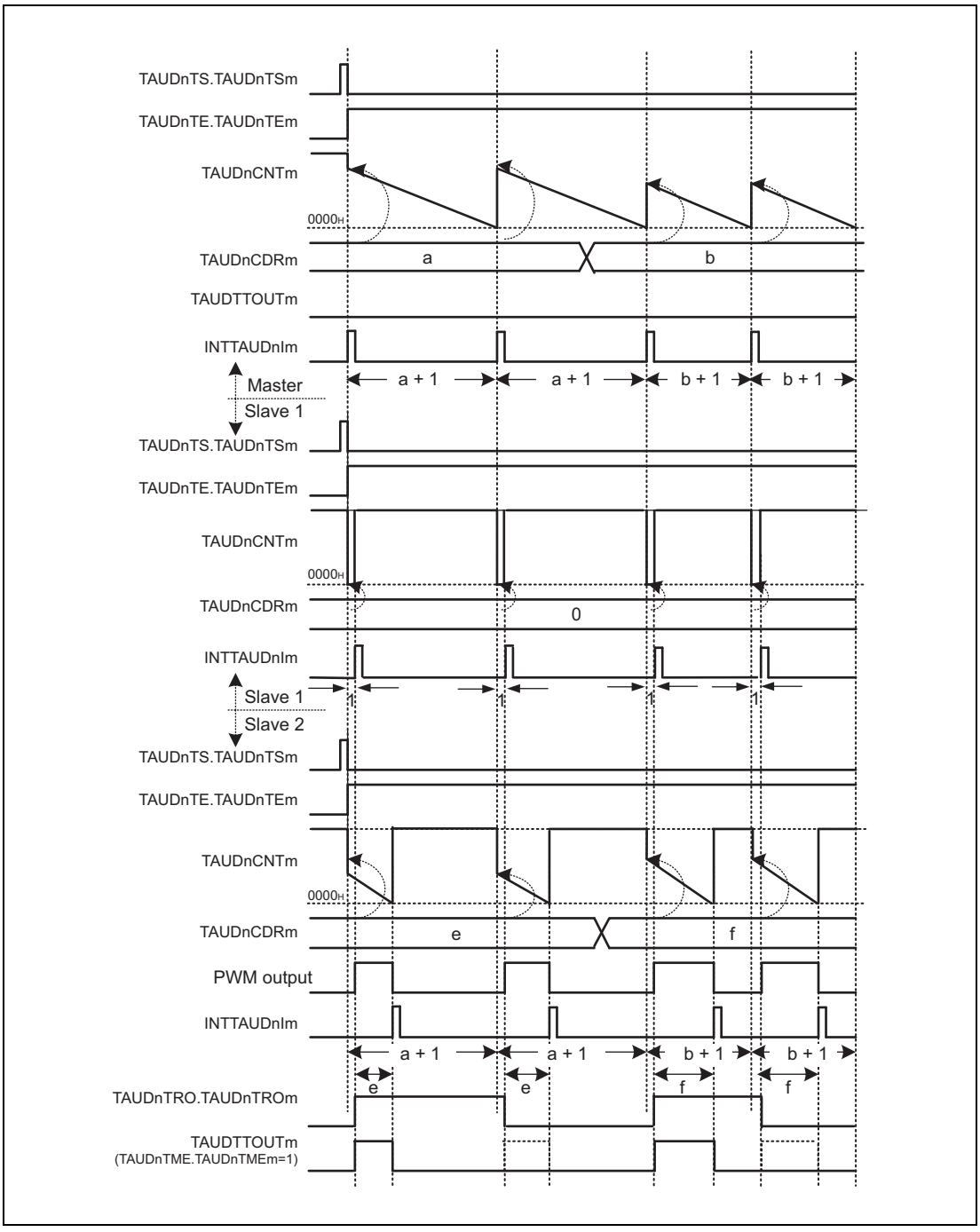



**Figure 25.130 General Timing Diagram of Non-Complementary Modulation Output Function Type 1**

**NOTE**

TAUDTTOUTm of slave channel 2 rises with a delay of one clock count after the rise of INTTAUDnIm of the master channel.

### 25.16.1.4 Register Settings for the Master Channel

#### (1) TAUDnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.220 Contents of the TAUDnCMORm Register for the Master Channel of Non-Complementary Modulation Output Function Type 1**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 1: Master channel |
| 10 to 8 | TAUDnSTS[2:0] | 000: Trigger the counter using software. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0000: Interval timer mode |
| 0 | TAUDnMD0 | 1: INTTAUDnIm is generated at the beginning of operation or at a restart time. |

#### (2) TAUDnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.221 Contents of the TAUDnCMURm Register for the Master Channel of Non-Complementary Modulation Output Function Type 1**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for the master channel

TAUDnTOE.TAUDnTOEm is set to 0 because channel output mode is not used with this function.

### (4) Simultaneous rewrite for the master channel

Both the master and slave channels should have the same simultaneous rewrite settings.

**Table 25.222 Simultaneous Rewrite Settings for the Master Channel of Non-Complementary Modulation Output Function Type 1**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Monitors master channel for simultaneous rewrite triggers.<br>1: Monitors upper channel other than the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 0: Generates a simultaneous rewrite trigger signal when the master channel starts to count. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. Monitors master channel for simultaneous rewrite triggers, regardless of the value of this bit, when TAUDnRDS.TAUDnRDSm = 0. |

**NOTE**

Use with TAUDnRDS.TAUDnRDSm bit = 1 requires an upper channel higher than the master channel that operates with **Section 25.14.1, Simultaneous Rewrite Trigger Generation Function Type 1**.

Conduct operation settings under the following conditions.

- Simultaneous rewrite trigger output function type 1 setting channel: TAUDnRDCm = 1, TAUDnRDSm = 1
  In addition, TAUDnCDRm settings for this channel are as follows.
  = ((TAUDnCDR setting for the master channel targeted for simultaneous rewrite + 1) × Interrupt count) – 1
- Master channel: TAUDnRDCm = 0, TAUDnRDSm = 1
- Slave channel: TAUDnRDCm = 0, TAUDnRDSm = 1

### 25.16.1.5 Register Settings for Slave Channel 1

#### (1) TAUDnCMORm for slave channel 1

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.223 Contents of the TAUDnCMORm Register for Slave Channel 1 of Non-Complementary Modulation Output Function Type 1**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 100: INTTAUDnIm of master channel is a start trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0100: One-count mode |
| 0 | TAUDnMD0 | 1: Start trigger during operation is valid. |

#### (2) TAUDnCMURm for slave channel 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.224 Contents of the TAUDnCMURm Register for Slave Channel 1 of Non-Complementary Modulation Output Function Type 1**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode

TAUDnTOE.TAUDnTOEm is set to 0 because the channel output mode is not used on slave channel 1 with this function. However, this mode can be used in independent channel output mode controlled by software.

**CAUTION**

TAUDnTRC.TAUDnTRCm should be set to 1 because slave channel 1 is used as a real-time output trigger channel.

### (4) Simultaneous rewrite for slave channel 1

Both the master and slave channels should have the same simultaneous rewrite settings.

**Table 25.225 Simultaneous Rewrite Settings for Slave Channel 1 of Non-Complementary Modulation Output Function Type 1**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Monitors master channel for simultaneous rewrite triggers.<br>1: Monitors upper channel other than the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 0: Generates a simultaneous rewrite trigger signal when the master channel starts to count. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. Monitors master channel for simultaneous rewrite triggers, regardless of the value of this bit, when TAUDnRDS.TAUDnRDSm = 0. |

### 25.16.1.6 Register Settings for Slave Channels 2 to 7

#### (1) TAUDnCMORm for slave channels 2 to 7

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.226 Contents of the TAUDnCMORm Register for Slave Channels 2 to 7 of Non-Complementary Modulation Output Function Type 1**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 100: INTTAUDnIm of master channel is a start trigger. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0100: One-count mode |
| 0 | TAUDnMD0 | 1: Start trigger during operation is valid. |

#### (2) TAUDnCMURm for slave channels 2 to 7

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.227 Contents of the TAUDnCMURm Register for Slave Channels 2 to 7 of Non-Complementary Modulation Output Function Type 1**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for slave channels 2 to 7

Table 25.228 Control Bit Settings in Synchronous Channel Output Mode 1 with Non-Complementary Modulation Output

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 1: Synchronous channel output |
| TAUDnTOC.TAUDnTOCm | 0: Operating mode 1 |
| TAUDnTOL.TAUDnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 1: Enables real-time output. |
| TAUDnTRO.TAUDnTROm | 0: Real-time output is low.<br>1: Real-time output is high. |
| TAUDnTRC.TAUDnTRCm | 0: Upper channel generates a real-time output trigger for channel m. |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation<br>1: Enables modulation |

### (4) Simultaneous rewrite of slave channels 2 to 7

Both the master and slave channels should have the same simultaneous rewrite settings.

Table 25.229 Simultaneous Rewrite Settings for Slave Channels 2 to 7 of Non-Complementary Modulation Output Function Type 1

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Monitors master channel for simultaneous rewrite triggers.<br>1: Monitors upper channel other than the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 0: Generates a simultaneous rewrite trigger signal when the master channel starts to count. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. Monitors master channel for simultaneous rewrite triggers, regardless of the value of this bit, when TAUDnRDS.TAUDnRDSm = 0. |

### 25.16.1.7 Operating Procedure for Non-Complementary Modulation Output Function Type 1

Table 25.230 Operating Procedure for Non-Complementary Modulation Output Function Type 1 (1/2)

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Master channel: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.16.1.4, Register Settings for the Master Channel**.<br><br>Slave channel 1: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.16.1.5, Register Settings for Slave Channel 1**.<br><br>Slave channels 2 to 7: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.16.1.6, Register Settings for Slave Channels 2 to 7**.<br><br>Set the value of TAUDnCDRm register of every channel. Set a pulse cycle with TAUDnCDRm of master channel, $0000_H$ in TAUDnCDRm of slave channel 1, and duty width with TAUDnCDRm of slave channels 2 to 7.<br><br>Set TAUDnTRC.TAUDnTRCm to 1 on slave channel 1. | Channel operation is stopped. |

**Table 25.230 Operating Procedure for Non-Complementary Modulation Output Function Type 1 (2/2)**

Restart Operation

| | Operation | TAUDn Status |
|---|---|---|
| Start Operation | Set TAUDnTS.TAUDnTSm of master and slave channels to 1 simultaneously.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm of master and slave channels is set to 1 and the counter starts counting down. |
| During Operation | TAUDnCDRm, TAUDnTRO.TAUDnTROm, and TAUDnTMEm.TAUDnTMEm can be changed at any time.<br>TAUDnCNTm and TAUDnRSF.TAUDnRSFm can be read at any time.<br><br>TAUDnRDT.TAUDnRDTm can be changed during operation. | TAUDnCDRm value of master channel, slave channel 1 and slave channels 2 to 7 is loaded into TAUDnCNTm to perform counting down. When the counter of master channel reaches $0000_H$:<br>• INTTAUDnIm is generated.<br>• TAUDnCDRm value of master channel is reloaded into TAUDnCNTm to continue counting down.<br>• PWM output signals of slave channels 2 to 7 are set.<br>• TAUDnCDRm value of slave channel 1 is reloaded into TAUDnCNTm to perform counting down.<br>• TAUDnCDRm value of slave channels 2 to 7 is reloaded into TAUDnCNTm to perform counting down.<br>• When the counter of slave channel 1 reaches $0000_H$:<br>– INTTAUDnIm is generated.<br>– The TAUDnTRO.TAUDnTROm value of slave channels 2 to 7 is reflected to the TAUDTTOUTm output.<br>• When the counter of slave channels 2 to 7 reaches $0000_H$:<br>– INTTAUDnIm is generated.<br>– PWM output signals of slave channels 2 to 7 are set.<br>TAUDTTOUTm of slave channels 2 to 7 outputs a PWM signal, a high-level signal or low-level signal depending on the values of real-time output bits (TAUDnTRO.TAUDnTROm) and modulation output bit (TAUDnTME.TAUDnTMEm) of a pair of slave channels. |
| Stop Operation | Set TAUDnTT.TAUDnTTm of master and slave channels to 1 simultaneously.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm and TAUDTTOUTm stop and retain their current values. |

### 25.16.1.8 Specific Timing Diagrams

The following settings apply to the specific timing diagram.

- Slave channels 2 to 7: Positive logic (TAUDnTOL.TAUDnTOLm = 0)

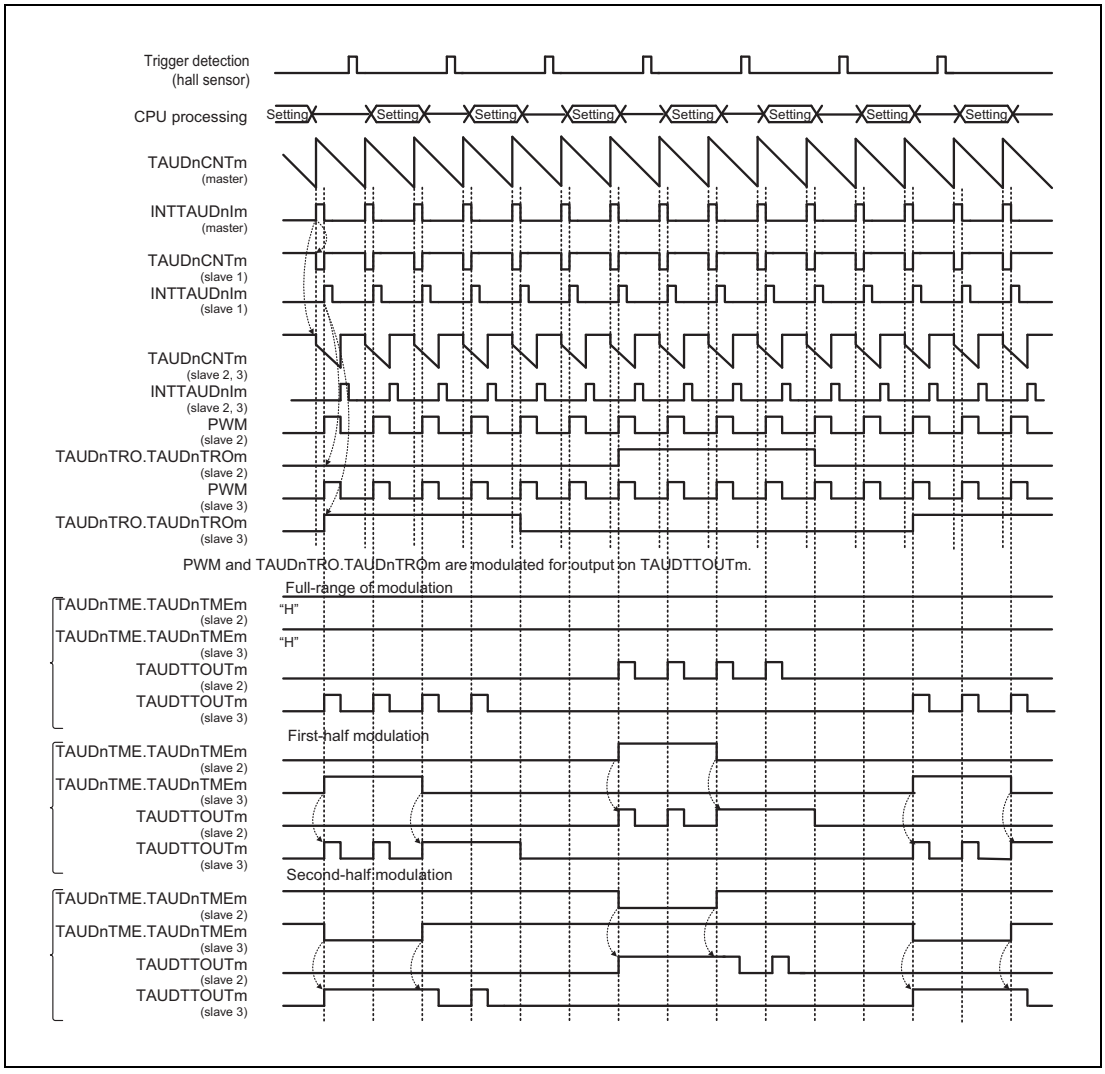



**Figure 25.131 Specific Timing Diagram of Non-Complementary Modulation Output Function Type 1**

The above timing diagram shows how full modulation, first-half modulation, and second-half modulation can be achieved by modifying the TAUDnTME.TAUDnTMEm bits of lower slave channels during operation.

The "Setting" symbol indicates a time period when the values of TAUDnCDRm, TAUDnTME.TAUDnTMEm, and TAUDnTRO.TAUDnTROm can be changed.

TAUDnTME.TAUDnTMEm setting is reflected by detecting the count start timing and master channel cycle. According to the modified setting, modulation waveforms are output from TAUDTTOUTm.

A TAUDnTRO.TAUDnTROm bit value is set by software, but a new setting is applied only when an interrupt occurs on slave channel 1.

## 25.16.2 Non-Complementary Modulation Output Function Type 2

### 25.16.2.1 Overview

#### Summary

This function outputs a triangular PWM output signal, a high-level signal, or low-level signal from TAUDTTOUTm depending on the real-time output bit value (TAUDnTRO.TAUDnTROm) and the modulation output enable bit value (TAUDnTME.TAUDnTMEm) of a pair of slave channels. Three pairs of channels are typically used.

#### Prerequisites

- One master channel and seven slave channels
- The operation mode of the master channel must be set to interval timer mode (See **Table 25.232, Contents of the TAUDnCMORm Register for the Master Channel of Non-Complementary Modulation Output Function Type 2**).
- The operating mode for slave channel 1 should be set to event count mode (See **Table 25.236, Contents of the TAUDnCMORm Register for Slave Channel 1 of Non-Complementary Modulation Output Function Type 2**).
- The operating mode for slave channels 2 to 7 should be set to count-up/-down mode (See **Table 25.239, Contents of the TAUDnCMORm Register for Slave Channels 2 to 7 of Non-Complementary Modulation Output Function Type 2**).
- The output mode for the master channel should be set to independent channel output mode 1. (See **Section 25.7, Channel Output Modes**.)
- This function does not use TAUDTTOUTm of slave channel 1 but TAUDnTRC.TAUDnTRCm should be set to 1 (See **Section 25.7, Channel Output Modes**).
- The channel output mode for slave channels 2 to 7 should be set to synchronous channel output mode 2 with non-complementary modulation output (See **Section 25.7, Channel Output Modes**).

### Functional description

The master/slave channel counter is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This sets TAUDnTE.TAUDnTEm = 1, enabling count operation. The value of data register (TAUDnCDRm) is loaded into the counter (TAUDnCNTm).

- Master channel:
  The counter of master channel starts to count down. When the counter reaches $0000_H$, INTTAUDnIm is generated.
- Slave channel 1:
  When slave channel 1 detects an interrupt from the master channel, the TAUDnCNTm value is decremented. When an interrupt from the master channel is detected (TAUDnCDRm + 1) times, INTTAUDnIm is generated. Then, the TAUDnCDRm value is loaded into TAUDnCNTm to continue operation subsequently.
  Since slave channel 1 is set as a real-time output trigger channel (TAUDnTRC.TAUDnTRCm = 1), if an interrupt occurs on slave channel 1, the real-time output bit (TAUDnTRO.TAUDnTROm) of the channel which monitors an interrupt on the corresponding channel is reflected to the TAUDTTOUTm output.
- Slave channel 2:
  Once detecting an interrupt from the master channel, TAUDnCNTm counts in the reverse direction. When an interrupt is detected during count-up operation, TAUDnCDRm value is reloaded and then the counter starts to count down.
  If TAUDnCNTm = $0001_H$, an interrupt occurs and a PWM output signal is set/reset.

The combined use of the master channel and slave channel 2 generates a PWM output signal. The master channel generates a PWM output cycle and slave channel 2 generate a duty cycle.

Slave channels 3 to 7 operate like slave channel 2.

A signal that is output from TAUDTTOUTm depends on a real-time output bit value (TAUDnTRO.TAUDnTROm) and a modulation output bit value (TAUDnTME.TAUDnTMEm) of the slave channel, as described in **Table 25.231, TAUDTTOUTm Output of Slave Channels in Non-Complementary Modulation Output Function Type 2 (TAUDnTOL.TAUDnTOLm = 0)**.

This function cannot make a forced restart. The counter can be stopped by setting TAUDnTT.TAUDnTTm of master and slave channels to 1. This sets TAUDnTE.TAUDnTEm to 0. TAUDnCNTm and TAUDTTOUTm of master and slave channels stop but retain their values. The counters can be restarted by setting TAUDnTS.TAUDnTSm to 1.

**Conditions**

- If TAUDnTME.TAUDnTMEm = 0 on slave channels 2 to 7 (TAUDnTOL.TAUDnTOLm = 0):
  - If the channel's TAUDnTRO.TAUDnTROm is set to 1, TAUDTTOUTm outputs a high-level signal.
  - If the channel's TAUDnTRO.TAUDnTROm is set to 0, TAUDTTOUTm outputs a low-level signal.
- If TAUDnTME.TAUDnTMEm = 1 on slave channels 2 to 7 (TAUDnTOL.TAUDnTOLm = 0):
  - If the channel's TAUDnTRO.TAUDnTROm is set to 1, TAUDTTOUTm outputs PWM (positive logic) corresponding to the channel.
  - If the channel's TAUDnTRO.TAUDnTROm is set to 0, TAUDTTOUTm outputs a low-level signal.
- If TAUDnTOL.TAUDnTOLm is set to 1, high-level and low-level signals output from TAUDTTOUTm are inverted. The PWM signal is negative logic. Only the initial setting of TAUDnTOL.TAUDnTOLm is permitted (cannot be changed during operation).

**Table 25.231 TAUDTTOUTm Output of Slave Channels in Non-Complementary Modulation Output Function Type 2 (TAUDnTOL.TAUDnTOLm = 0)**

| TAUDnTME.TAUDnTMEm | TAUDnTRO.TAUDnTROm | TAUDTTOUTm Output |
|---|---|---|
| 0 | 0 | Low level |
| | 1 | High level |
| 1 | 0 | Low level |
| | 1 | PWM (positive logic) |

- This function enables simultaneous rewrite. See **Section 25.6, Simultaneous Rewrite**.
- If TAUDnTOL.TAUDnTOLm is set to 0 on slave channels 2 to 7, TAUDnTO.TAUDnTOm is set to 0 (low) before TAUDnTE.TAUDnTEm is set to 0.
- If TAUDnTOL.TAUDnTOLm is set to 1 on slave channels 2 to 7, TAUDnTO.TAUDnTOm is set to 1 (high) before TAUDnTE.TAUDnTEm is set to 0.

### 25.16.2.2 Equations

Slave channels 2 to 7:

Carrier cycle (down/up) = [TAUDnCDRm (master) + 1] × 2 × count clock cycle

Duty time = [TAUDnCDRm (master) + 1 - TAUDnCDRm (slave)] × 2 × count clock cycle

### 25.16.2.3 Block Diagram and General Timing Diagram

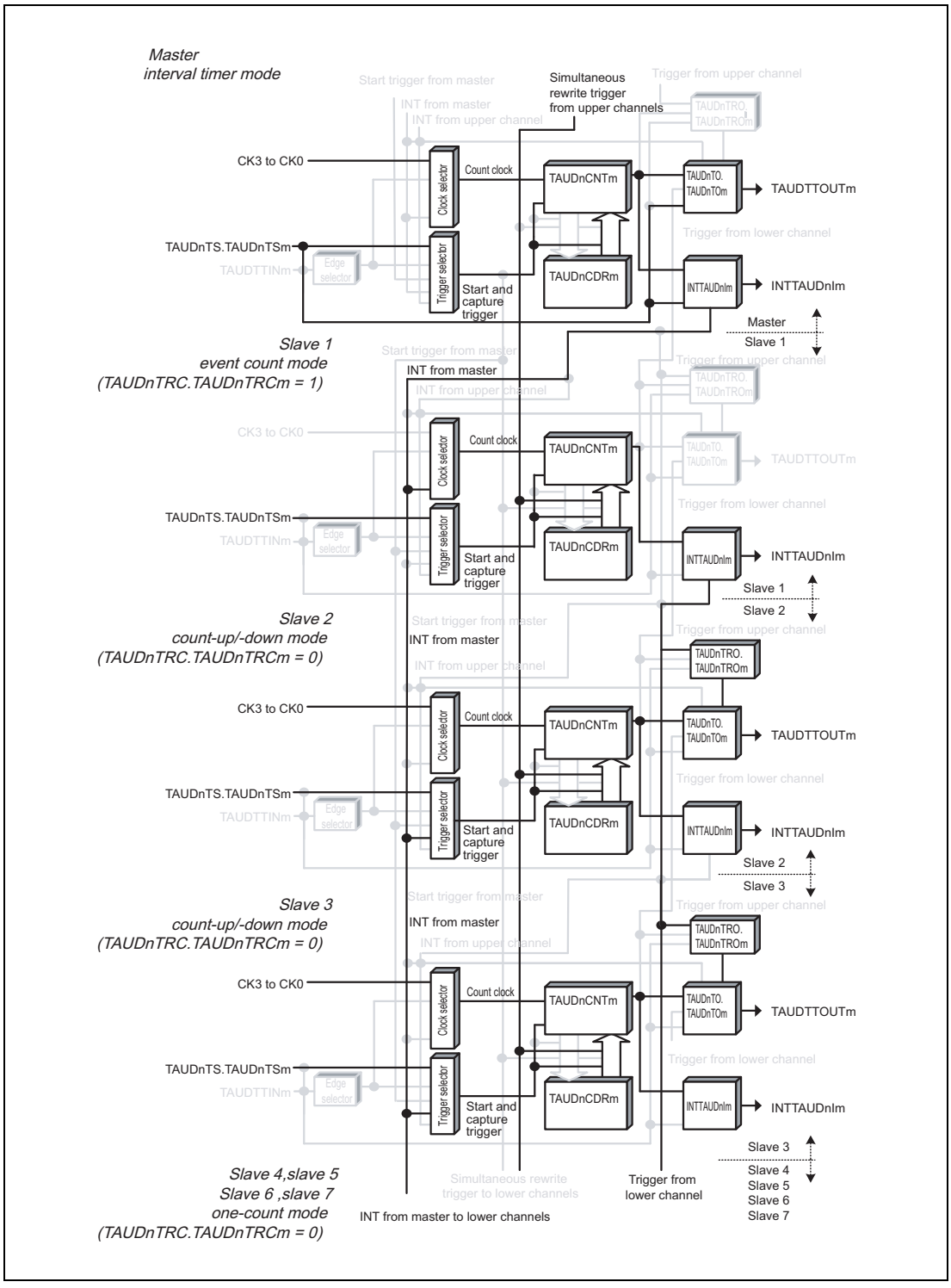



**Figure 25.132 Block Diagram of Non-Complementary Modulation Output Function Type 2**

The following settings apply to the general timing diagram.

- Master channel: INTTAUDnIm is not generated at the beginning of operation. (TAUDnCMORm.TAUDnMD0 = 0)
- Slave channels 2 to 7: Positive logic (TAUDnTOL.TAUDnTOLm = 0)

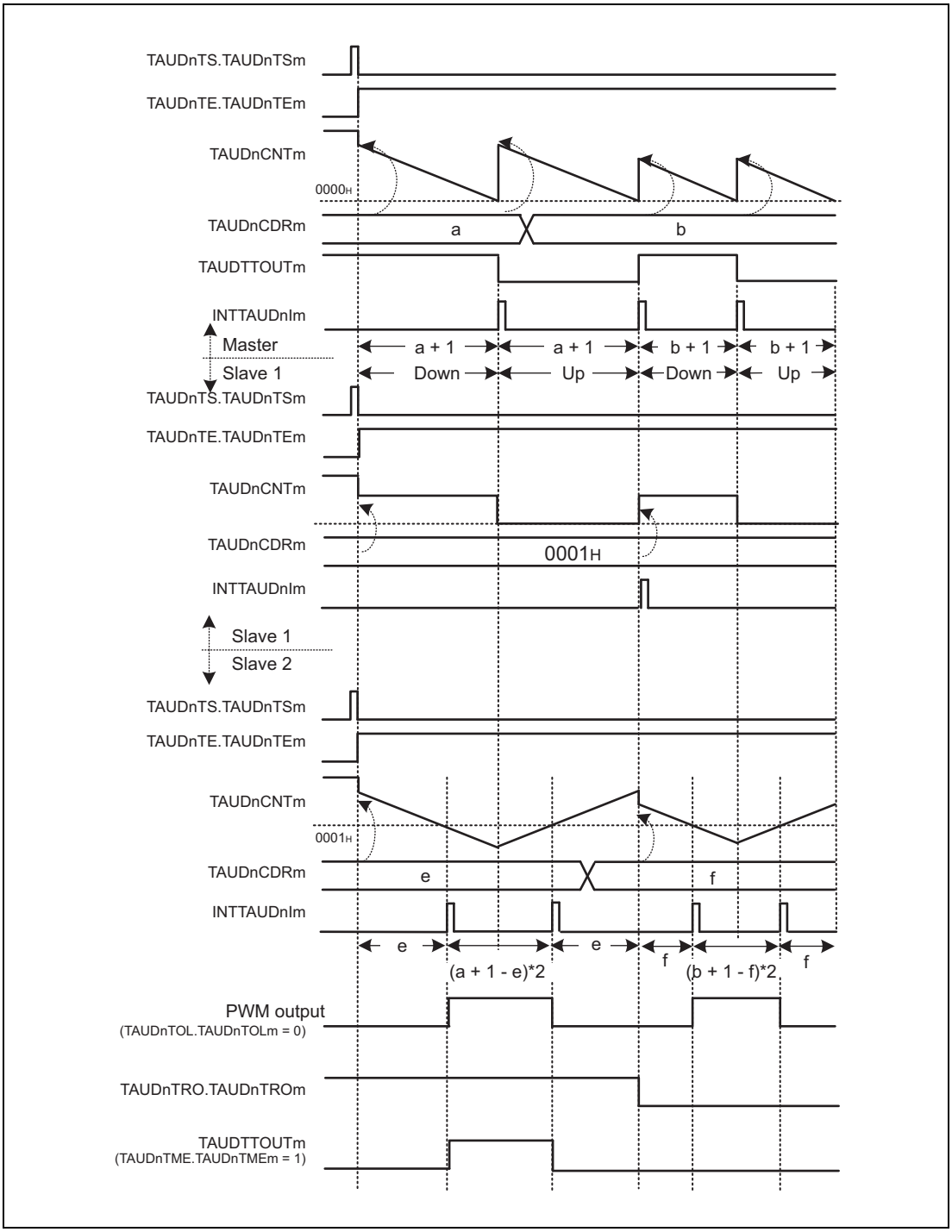



**Figure 25.133 General Timing Diagram of Non-Complementary Modulation Output Function Type 2**

### 25.16.2.4 Register Settings for the Master Channel

#### (1) TAUDnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.232 Contents of the TAUDnCMORm Register for the Master Channel of Non-Complementary Modulation Output Function Type 2**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 1: Master channel |
| 10 to 8 | TAUDnSTS[2:0] | 000: Trigger the counter using software. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0000: Interval timer mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm is not generated at the beginning of operation or at a restart time.<br>1: INTTAUDnIm is generated at the beginning of operation or at a restart time. |

#### (2) TAUDnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.233 Contents of the TAUDnCMURm Register for the Master channel of Non-Complementary Modulation Output Function Type 2**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for the master channel

Table 25.234 Control Bit Settings for the Master Channel in Non-Complementary Modulation Output Function Type 2

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 0: Independent channel output |
| TAUDnTOC.TAUDnTOCm | 0: Operating mode 1 (toggle mode with TAUDnTOM.TAUDnTOMm = 0) |
| TAUDnTOL.TAUDnTOLm | 0: The setting is disabled in toggle mode (the value after reset). |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set these bits to 0 |
| TAUDnTRC.TAUDnTRCm | |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

### (4) Simultaneous rewrite for the master channel

Both the master and slave channels should have the same simultaneous rewrite settings.

Table 25.235 Simultaneous Rewrite Settings for the Master Channel of Non-Complementary Modulation Output Function Type 2

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite |
| TAUDnRDS.TAUDnRDSm | 0: Monitors master channel for simultaneous rewrite triggers.<br>1: Monitors upper channel other than the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 1: A simultaneous rewrite trigger signal is generated when master channel starts to count and the corresponding slave channel is at the peak of a triangular wave. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. |

**NOTE**

If TAUDnRDS.TAUDnRDSm = 1, it is necessary for an upper channel higher than the master channel to generate a simultaneous rewrite trigger signal.

### 25.16.2.5 Register Settings for Slave Channel 1

#### (1) TAUDnCMORm for slave channel 1

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.236 Contents of the TAUDnCMORm Register for Slave Channel 1 of Non-Complementary Modulation Output Function Type 2**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 11: INTTAUDnIm of the master channel is used as the count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 000: Trigger the counter using software.<br>011: Triggers simultaneous rewrite. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0011: Event count mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm is not generated at the beginning of operation or at a restart time. |

#### (2) TAUDnCMURm for slave channel 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.237 Contents of the TAUDnCMURm Register for Slave Channel 1 of Non-Complementary Modulation Output Function Type 2**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode

TAUDnTOE.TAUDnTOEm is set to 0 because the channel output mode is not used on slave channel 1 with this function. However, this mode can be used in independent channel output mode controlled by software.

**CAUTION**

TAUDnTRC.TAUDnTRCm should be set to 1 because slave channel 1 is used as a real-time output trigger channel.

### (4) Simultaneous rewrite for slave channel 1

Both the master and slave channels should have the same simultaneous rewrite settings.

**Table 25.238 Simultaneous Rewrite Settings for Slave Channel 1 of Non-Complementary Modulation Output Function Type 2**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Monitors master channel for simultaneous rewrite triggers.<br>1: Monitors upper channel other than the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 1: Simultaneous rewrite trigger signal is generated when master channel counter is started and the corresponding slave channel is at the peak of triangular wave. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. Monitors master channel for simultaneous rewrite triggers, regardless of the value of this bit, when TAUDnRDS.TAUDnRDSm = 0. |

### 25.16.2.6 Register settings for slave channels 2 to 7

#### (1) TAUDnCMORm for slave channels 2 to 7

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.239 Contents of the TAUDnCMORm Register for Slave Channels 2 to 7 of Non-Complementary Modulation Output Function Type 2**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 111: The up/down output trigger signal of the master channel |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 1001: Count-up/-down mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm is not generated at the beginning of operation or at a restart time. |

#### (2) TAUDnCMURm for slave channels 2 to 7

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.240 Contents of the TAUDnCMURm Register for Slave Channels 2 to 7 of Non-Complementary Modulation Output Function Type 2**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Output mode for slave channels 2 to 7

Table 25.241 Control Bit Settings in Synchronous Channel Output Mode 2 with Non-Complementary Modulation Output

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 1: Synchronous channel output |
| TAUDnTOC.TAUDnTOCm | 1: Operating mode 2 |
| TAUDnTOL.TAUDnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 1: Enables real-time output. |
| TAUDnTRO.TAUDnTROm | 0: Real-time output is low.<br>1: Real-time output is high. |
| TAUDnTRC.TAUDnTRCm | 0: The upper channel generates the real-time output trigger for channel m |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation<br>1: Enables modulation |

### (4) Simultaneous rewrite for slave channels 2 to 7

Both the master and slave channels should have the same simultaneous rewrite settings.

Table 25.242 Simultaneous Rewrite Settings for Slave Channels 2 to 7 of Non-Complementary Modulation Output Function Type 2

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Monitors master channel for simultaneous rewrite triggers.<br>1: Monitors upper channel other than the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 1: Simultaneous rewrite trigger signal is generated when master channel counter is started and the corresponding slave channel is at the peak of triangular wave. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. Monitors master channel for simultaneous rewrite triggers, regardless of the value of this bit, when TAUDnRDS.TAUDnRDSm = 0. |

### 25.16.2.7 Operating Procedure for Non-Complementary Modulation Output Function Type 2

Table 25.243 Operating Procedure for Non-Complementary Modulation Output Function Type 2 (1/2)

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Master channel: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.16.2.4, Register Settings for the Master Channel**.<br><br>Slave channel 1: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.16.2.5, Register Settings for Slave Channel 1**.<br><br>Slave channels 2 to 7: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.16.2.6, Register settings for slave channels 2 to 7**.<br><br>Set the value of TAUDnCDRm register of every channel. Set pulse cycle in TAUDnCDRm of master channel, and in TAUDnCDRm of slave channel 1, set the number of interrupts from master channel to be ignored before slave channel 1 generates a real-time output trigger. Set duty width in TAUDnCDRm of slave channels 2 to 7.<br><br>Set TAUDnTRC.TAUDnTRCm to 1 on slave channel 1. | Channel operation is stopped. |

**Table 25.243 Operating Procedure for Non-Complementary Modulation Output Function Type 2 (2/2)**

| | Operation | TAUDn Status |
|---|---|---|
| Start Operation | Set TAUDnTS.TAUDnTSm of master and slave channels to 1 simultaneously.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm of master and slave channels is set to 1 and the counter starts counting down. |
| During Operation | TAUDnCDRm, TAUDnTRO.TAUDnTROm, and TAUDnTMEm.TAUDnTMEm can be changed at any time.<br>TAUDnCNTm and TAUDnRSF.TAUDnRSFm can be read at any time.<br><br>TAUDnRDT.TAUDnRDTm can be changed during operation. | The TAUDnCDRm value of master channel and slave channels 2 to 7 is loaded into TAUDnCNTm to perform counting down. The TAUDnCDRm value of slave channel 1 is loaded and the counter waits for an interrupt from the master channel. When the counter of master channel reaches $0000_H$:<br>• INTTAUDnIm is generated.<br>• TAUDnCDRm value is reloaded into TAUDnCNTm to continue counting down.<br>• The TAUDnCNTm value of slave channel 1 decrements by 1 and the counter waits for a next interrupt from the master channel.<br>• TAUDnCNTm of slave channels 2 to 7 reloads the TAUDnCDRm value, or performs counting in opposite direction.<br>• At the same timing when the TAUDnCDRm value is loaded, the TAUDnTME.TAUDnTMEm value of slave channels 2 to 7 is reflected to the TAUDTTOUTm output.<br>• When slave channel 1 detects an interrupt from the master channel for the (TAUDnCDRm + 1) times:<br>– INTTAUDnIm is generated.<br>– The TAUDnTRO.TAUDnTROm value of slave channels 2 to 7 is reflected to the TAUDTTOUTm output.<br>• When the counter of slave channels 2 to 7 reaches $0001_H$:<br>– INTTAUDnIm is generated.<br>– PWM output signals of slave channels 2 to 7 are set/reset. |
| Stop Operation | Set TAUDnTT.TAUDnTTm of master and slave channels to 1 simultaneously.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm and TAUDTTOUTm stop and retain their current values. |

Restart Operation

### 25.16.2.8 Specific Timing Diagrams

The following settings apply to the general timing diagram.

- Master channel: INTTAUDnIm is not generated at the beginning of operation. (TAUDnCMORm.TAUDnMD0 = 0)
- Slave channels 2 to 7: Positive logic (TAUDnTOL.TAUDnTOLm = 0)

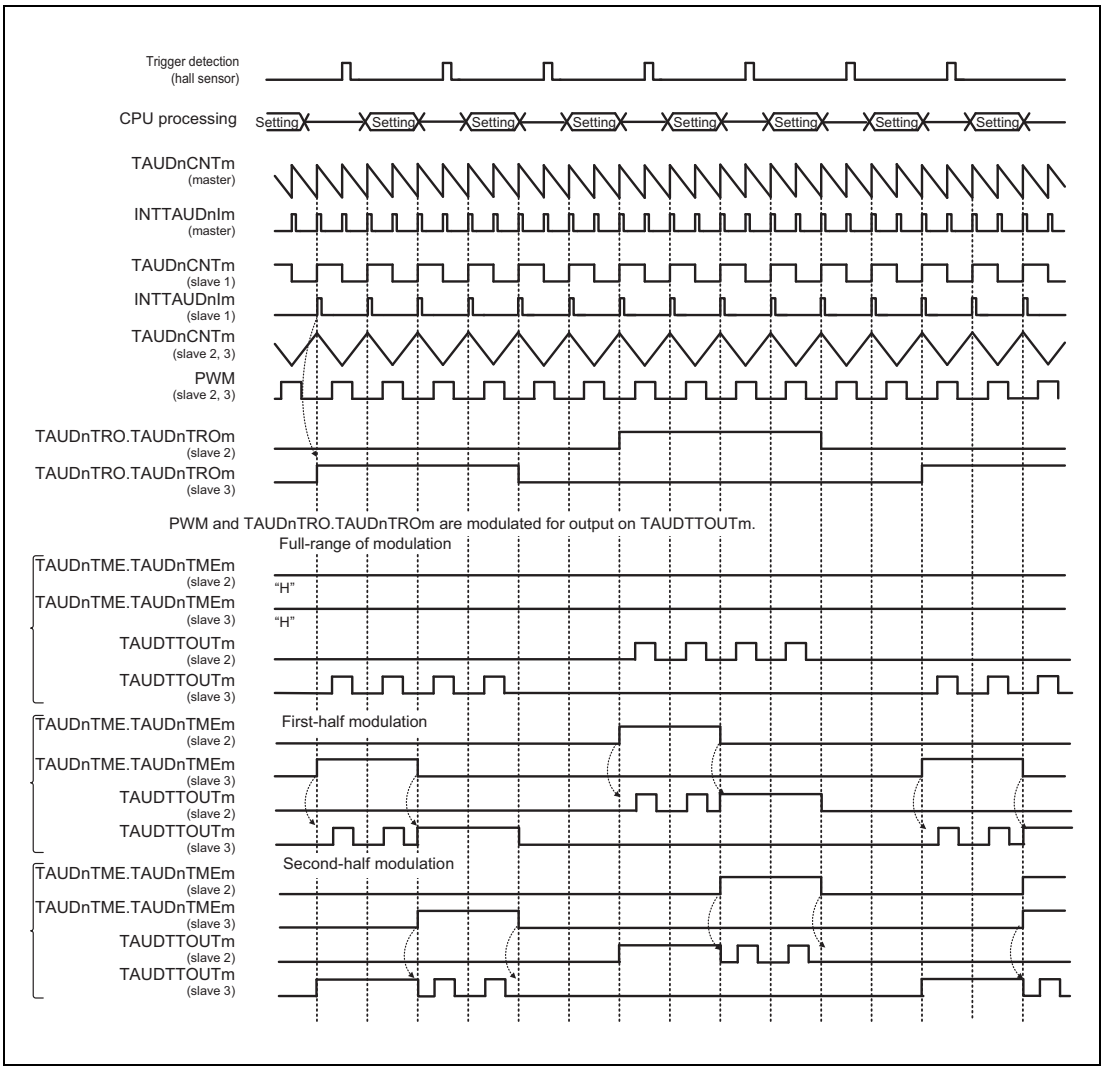



**Figure 25.134 Specific Timing Diagram of Non-Complementary Modulation Output Function Type 2**

The above timing diagram shows how full modulation, first-half modulation, and second-half modulation can be achieved by modifying the TAUDnTME.TAUDnTMEm bits of lower slave channels during operation.

The "Setting" symbol indicates a time period when the values of TAUDnCDRm, TAUDnTME.TAUDnTMEm, and TAUDnTRO.TAUDnTROm can be changed.

TAUDnTME.TAUDnTMEm setting is reflected by detecting the count start timing and triangle PWM carrier cycle (peak interrupt timing).

TAUDnTRO.TAUDnTROm bit value is set by software, but a new setting is applied only when an interrupt occurs on slave channel 1.

## 25.16.3 Complementary Modulation Output Function

### 25.16.3.1 Overview

#### Summary

This function outputs a triangle PWM output signal, a high-level signal, or low-level signal from TAUDTTOUTm with dead time added, depending on the real-time output bit value (TAUDnTRO.TAUDnTROm) and the modulation output bit value (TAUDnTME.TAUDnTMEm) of a pair of slave channels, and an output level bit value (TAUDnTDL.TAUDnTDLm). Three pairs of channels are typically used.

#### Prerequisites

- One master channel and seven slave channels
- The operation mode of the master channel must be set to interval timer mode (See **Table 25.245, Contents of the TAUDnCMORm Register for the Master Channel of the Complementary Modulation Output Function**).
- The operating mode for slave channel 1 should be set to event count mode (See **Table 25.249, Contents of the TAUDnCMORm Register for Slave Channel 1 of the Complementary Modulation Output Function**).
- The operating mode for slave channels 2, 4 and 6 should be set to count-up/-down mode (See **Table 25.252, Contents of the TAUDnCMORm Register for Slave Channel 2, 4, and 6 of the Complementary Modulation Output Function**).
- The operating mode for slave channels 3, 5 and 7 should be set to one-count mode (See **Table 25.256, Contents of the TAUDnCMORm Register for Slave Channel 3, 5, and 7 of the Complementary Modulation Output Function**).
  In addition, as the number of occurrences of an interrupt for slave channels 3, 5 and 7 within the carrier cycle is not uniquely determined, do not use the interrupt as an interrupt source.
- The output mode for master channels should be set to independent channel output mode 1 (See **25.7, Channel Output Modes**).
- This function does not use TAUDTTOUTm of slave channel 1 but TAUDnTRC.TAUDnTRCm should be set to 1 (See **Section 25.7, Channel Output Modes**).
- The channel output mode for slave channels 2 to 7 should be set to synchronous channel output mode 2 with complementary modulation output (See **Section 25.7, Channel Output Modes**).

#### Functional description

- Master channel:
  The counter of the master channel is enabled by setting the channel trigger bit (TAUDnTS.TAUDnTSm) to 1. This sets TAUDnTE.TAUDnTEm = 1, enabling count operation. The value of data register (TAUDnCDRm) of the master channel is loaded into the counter (TAUDnCNTm) and the counter starts to count down from this value.
  When the counter of master channel reaches $0000_H$, INTTAUDnIm is generated. This decrements the counter value of slave channel 1 by 1 and the counter of slave channel 2 starts to count in the opposite direction.
- Slave channel 1:
  When the counter reaches $0000_H$, slave channel 1 waits for the next interrupt from the master

channel. And the TAUDnCDRm value is reloaded into TAUDnCNTm (slave 1) and INTTAUDnIm is generated.
Slave channel 1 is set as a real-time output trigger channel (TAUDnTRC.TAUDnTRCm = 1). The value of real-time output bit (TAUDnTRO.TAUDnTROm) of each channel is applied to the channel that detects the occurrence of an interrupt on slave channel 1 by an interrupt. The real-time output bit value can be changed in any timing by application software but a new value is not applied until an interrupt occurs on slave channel 1.

- Slave channel 2:
  When the slave channel 2 counter reaches $0001_H$, the slave channel 3 counter starts counting down. When the slave channel 3 counter reaches $0000_H$, an interrupt occurs.
- Slave channels 2 and 3:
  The combined use of the master channel and slave channels 2 and 3 generates a PWM output signal. The master channel generates a PWM output cycle, slave channel 2 generates a duty cycle, and slave channel 3 generates dead time.
- Slave channels 4 to 7:
  Slave channels 4 and 6 operate like slave channel 2. Slave channels 5 and 7 operate like slave channel 3.

A signal that is output from TAUDTTOUTm depends on a real-time output bit value (TAUDnTRO.TAUDnTROm), a modulation output bit value (TAUDnTME.TAUDnTMEm), and an output level bit value (TAUDnTDL.TAUDnTDLm) of the slave channel, as described in **Table 25.244, TAUDTTOUTm Output (TAUDnTOL.TAUDnTOLm = 0) for a Pair of Slave Channels of Complementary Modulation Output Function**.

It is, however, prohibited that a high-level signal is output from both channel 2 and channel 3 (in order to prevent a motor driver short circuit).

Forced restart is not possible for this function. The counter can be stopped by setting TAUDnTT.TAUDnTTm of master and slave channels to 1. This sets TAUDnTE.TAUDnTEm to 0. TAUDnCNTm and TAUDTTOUTm of master and slave channels stop but retain their values. The counters can be restarted by setting TAUDnTS.TAUDnTSm to 1.

#### Conditions

- If TAUDnTME.TAUDnTMEm of a pair of channels is set to 1 (TAUDnTOL.TAUDnTOLm = 0):
  - If TAUDnTRO.TAUDnTROm of one channel is set to 1, TAUDTTOUTm outputs the corresponding PWM of the channel.
  - If TAUDnTRO.TAUDnTROm of both channels is set to 0, TAUDTTOUTm of a pair outputs a low-level signal.
- If TAUDnTME.TAUDnTMEm of a pair of channels is set to 0 (TAUDnTOL.TAUDnTOLm = 0):
  - If TAUDnTRO.TAUDnTROm is set to 1, TAUDTTOUTm of the channel outputs a high-level signal.
  - If TAUDnTRO.TAUDnTROm is set to 0, TAUDTTOUTm of the channel outputs a low-level signal.
- If TAUDnTOL.TAUDnTOLm is set to 1, high-level and low-level signals output from TAUDTTOUTm are inverted. The PWM signal is negative logic.

**Table 25.244 TAUDTTOUTm Output (TAUDnTOL.TAUDnTOLm = 0) for a Pair of Slave Channels of Complementary Modulation Output Function**

| TAUDnTME.TAUDnTME02 | TAUDnTME.TAUDnTME03 | TAUDnTRO.TAUDnTRO02 | TAUDnTRO.TAUDnTRO03 | TAUDnTDL.TAUDnTDL02 | TAUDnTDL.TAUDnTDL03 | TAUDTTOUT2 Output | TAUDTTOUT3 Output |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | X | X | Low level | Low level |
| | | 0 | 1 | 1 | 0 | Low level | High level |
| | | 1 | 0 | 0 | 1 | High level | Low level |
| | | 1 | 1 | X | X | Setting prohibited | Setting prohibited |
| 1 | 1 | 0 | 0 | X | X | Low level | Low level |
| | | 0 | 1 | 1 | 0 | ~PWM | PWM |
| | | 1 | 0 | 0 | 1 | PWM | ~PWM |
| | | 1 | 1 | X | X | Setting prohibited | Setting prohibited |

**NOTES**

1. In the above table, PWM indicates a positive PWM signal and ~PWM indicates an inverted PWM signal (positive logic). PWM and ~PWM are set by TAUDnTDL.TAUDnTDLm.
2. Any settings not listed above are prohibited.

- If TAUDnTME.TAUDnTMEm is continuously set to 1 while TAUDnTRO.TAUDnTROm of one of paired channels is set to 1, full modulation is applied.
- If TAUDnTME.TAUDnTMEm is set to 1 at the first half of the period while TAUDnTRO.TAUDnTROm of one of paired channels is set to 1, first-half modulation is applied.
- If TAUDnTME.TAUDnTMEm is set to 1 at the second half of the period while TAUDnTRO.TAUDnTROm of one of paired channels is set to 1, second-half modulation is applied.
- Whether dead time is added to a normal or reverse phase PWM signal when two channels become high-level signal outputs simultaneously depends on a TAUDnTDL.TAUDnTDLm bit value.
  - If TAUDnTDL.TAUDnTDLm = 0, dead time is added to a normal phase PWM signal.
  - If TAUDnTDL.TAUDnTDLm = 1, dead time is added to a reverse phase PWM signal.
  - The operation defined by a TAUDnTDL.TAUDnTDLm bit value should be conducted by application software during operation. To modify TAUDnTDL.TAUDnTDLm, rewrite it during the period when TAUDnTRO.TAUDnTROm is $00_B$.
- The TAUDnCDRm value of slave channel 1 should be set to the value to generate INTTAUDnIm of slave channel 1 at a carrier cycle (peak interrupt timing).
- If TAUDnTOL.TAUDnTOLm is set to 0 on slave channels 2 to 7:
  - If TAUDnTDL.TAUDnTDLm is set to 0, TAUDnTO.TAUDnTOm is set to 0 (low) before TAUDnTE.TAUDnTEm is set to 0.
  - If TAUDnTDL.TAUDnTDLm is set to 1, TAUDnTO.TAUDnTOm is set to 1 (high) before TAUDnTE.TAUDnTEm is set to 0.
- If TAUDnTOL.TAUDnTOLm is set to 1 on slave channels 2 to 7:
  - If TAUDnTDL.TAUDnTDLm is set to 0, TAUDnTO.TAUDnTOm is set to 1 (high) before TAUDnTE.TAUDnTEm is set to 0.
  - If TAUDnTDL.TAUDnTDLm is set to 1, TAUDnTO.TAUDnTOm is set to 0 (low) before TAUDnTE.TAUDnTEm is set to 0.

- This function enables simultaneous rewrite. See **Section 25.6, Simultaneous Rewrite**.

#### 25.16.3.2 Equations

Pulse period = (TAUDnCDRm (master) + 1) × count clock cycle

$0000_H \leq$ TAUDnCDRm (master) $< FFFF_H$

Carrier cycle (down/up) = (TAUDnCDRm (master) + 1) × 2 × count clock cycle

For slave channels 2 and 3:

PWM signal width (positive phase) = [ (TAUDnCDRm (master) + 1 – TAUDnCDRm (slave 2) × 2) – (TAUDnCDRm (slave 3) + 1) ] × count clock cycle

PWM signal width (negative phase) = [ (TAUDnCDRm (master) + 1 – TAUDnCDRm (slave 2) × 2) + (TAUDnCDRm (slave 3) + 1) ] × count clock cycle

For slave channels 4 to 7:

Slave channels 4 and 6 are calculated in the same way as slave channel 2, whereas slave channels 5 and 7 are calculated as slave channel 3.

### 25.16.3.3 Block Diagram and General Timing Diagram

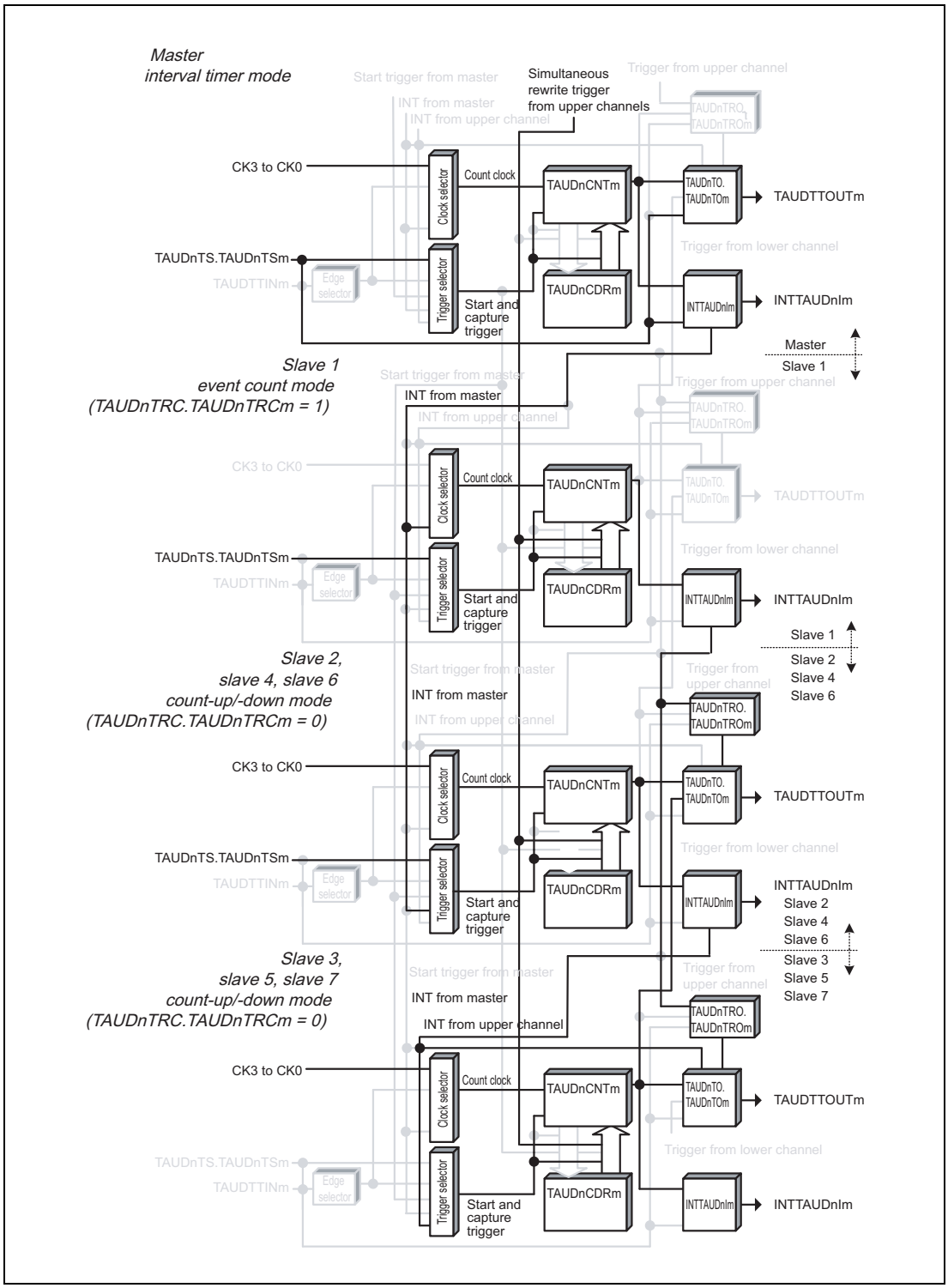



**Figure 25.135 Block Diagram of Complementary Modulation Output Function**

The following settings apply to the general timing diagram.

- Master channel: INTTAUDnIm is not generated at the beginning of operation. (TAUDnCMORm.TAUDnMD0 = 0)
- Slave channel 1: TAUDnCDRm = $0001_H$

- Slave channels 2 to 7: Positive logic (TAUDnTOL.TAUDnTOLm = 0)

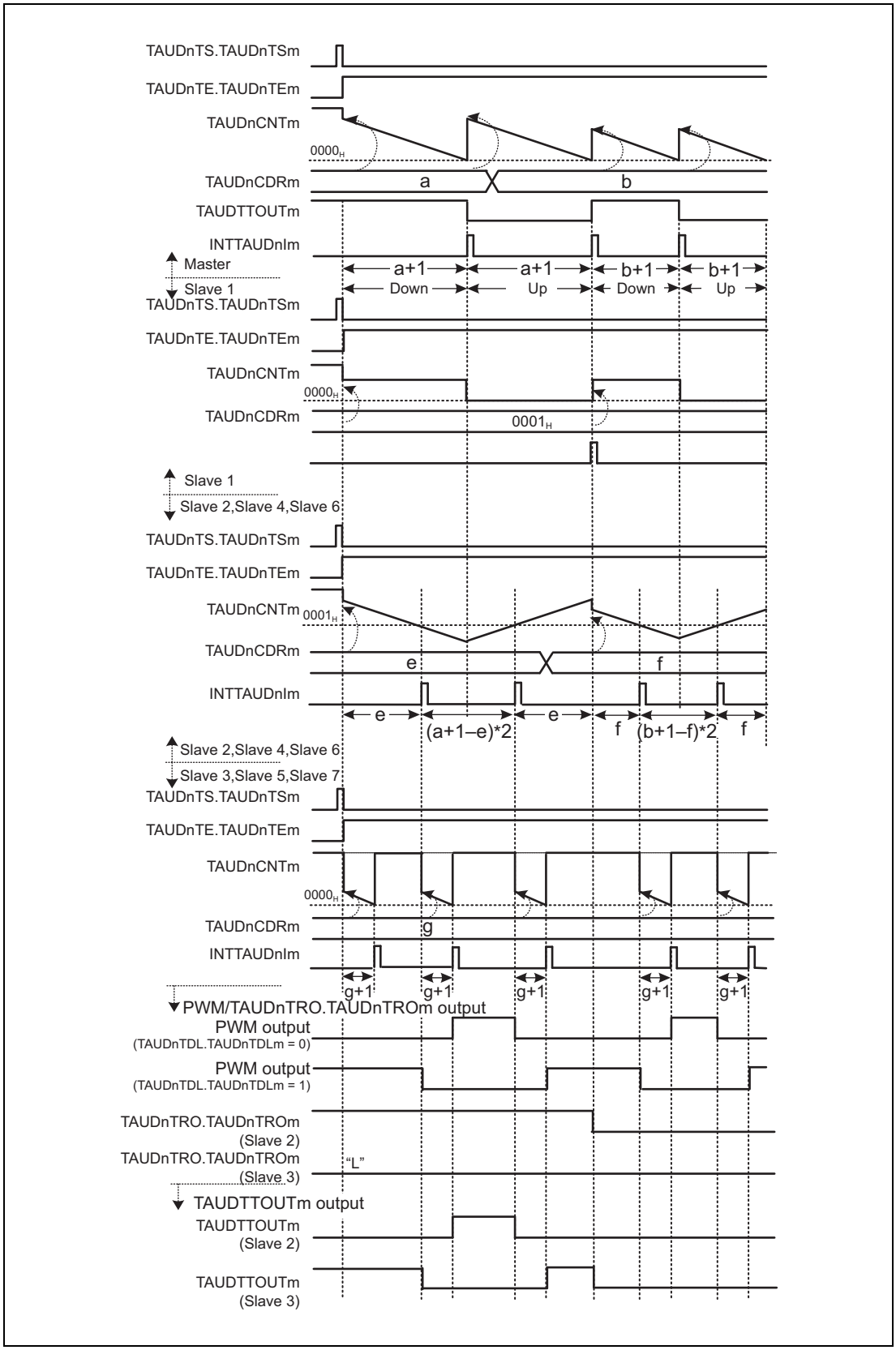



**Figure 25.136 General Timing Diagram of Complementary Modulation Output Function**

### 25.16.3.4 Register Settings for the Master Channel

#### (1) TAUDnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.245 Contents of the TAUDnCMORm Register for the Master Channel of the Complementary Modulation Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 1: Master channel |
| 10 to 8 | TAUDnSTS[2:0] | 000: Trigger the counter using software. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0000: Interval timer mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm is not generated and TAUDTTOUTm is not toggled at the beginning of operation or at a restart time.<br>1: INTTAUDnIm is generated to toggle TAUDTTOUTm at the beginning of an operation. |

#### (2) TAUDnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.246 Contents of the TAUDnCMURm Register for the Master Channel of the Complementary Modulation Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode for the master channel

**Table 25.247 Control Bit Settings in Independent Channel Output Mode 1**

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 0: Independent channel output |
| TAUDnTOC.TAUDnTOCm | 0: Operating mode 1 (Toggle mode if TAUDnTOM.TAUDnTOMm = 0) |
| TAUDnTOL.TAUDnTOLm | 0: The setting is disabled in toggle mode (the value after reset). |
| TAUDnTDE.TAUDnTDEm | 0: Disables dead time operation |
| TAUDnTDM.TAUDnTDMm | 0: When dead time operation is disabled (TAUDnTDE.TAUDnTDEm = 0), set these bits to 0 |
| TAUDnTDL.TAUDnTDLm | |
| TAUDnTRE.TAUDnTREm | 0: Disables real-time output |
| TAUDnTRO.TAUDnTROm | 0: When real-time output is disabled (TAUDnTRE.TAUDnTREm = 0), set these bits to 0 |
| TAUDnTRC.TAUDnTRCm | |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation |

### (4) Simultaneous rewrite for the master channel

Both the master and slave channels should have the same simultaneous rewrite settings.

**Table 25.248 Simultaneous Rewrite Settings for the Master Channel of Complementary Modulation Output Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Monitors master channel for simultaneous rewrite triggers.<br>1: Monitors upper channel other than the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 1: Simultaneous rewrite trigger signal is generated when master channel counter is started and the corresponding slave channel is at the peak of triangular wave. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. Monitors master channel for simultaneous rewrite triggers, regardless of the value of this bit, when TAUDnRDS.TAUDnRDSm = 0. |

**NOTE**

If TAUDnRDS.TAUDnRDSm = 1, it is necessary for an upper channel higher than the master channel to generate a simultaneous rewrite trigger signal.

### 25.16.3.5 Register Settings for Slave Channel 1

#### (1) TAUDnCMORm for slave channel 1

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.249 Contents of the TAUDnCMORm Register for Slave Channel 1 of the Complementary Modulation Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 11: INTTAUDnIm of the master channel is used as the count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 000: Trigger the counter using software.<br>011: Triggers simultaneous rewrite. |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0011: Event count mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm is not generated at the beginning of operation or at a restart time. |

#### (2) TAUDnCMURm for slave channel 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.250 Contents of the TAUDnCMURm Register for Slave Channel 1 of the Complementary Modulation Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Channel output mode

TAUDnTOE.TAUDnTOEm is set to 0 because the channel output mode is not used on slave channel 1 with this function. However, this mode can be used in independent channel output mode controlled by software.

**CAUTION**

TAUDnTRC.TAUDnTRCm should be set to 1 because slave channel 1 is used as a real-time output trigger channel.

### (4) Simultaneous rewrite for slave channel 1

Both the master and slave channels should have the same simultaneous rewrite settings.

**Table 25.251 Simultaneous Rewrite Settings for Slave Channel 1 of Complementary Modulation Output Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Monitors master channel for simultaneous rewrite triggers.<br>1: Monitors upper channel other than the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 1: Simultaneous rewrite trigger signal is generated when master channel counter is started and the corresponding slave channel is at the peak of triangular wave. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. Monitors master channel for simultaneous rewrite triggers, regardless of the value of this bit, when TAUDnRDS.TAUDnRDSm = 0. |

### 25.16.3.6 Register settings for slave channels 2, 4, and 6

#### (1) TAUDnCMORm for slave channels 2, 4, and 6

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.252 Contents of the TAUDnCMORm Register for Slave Channel 2, 4, and 6 of the Complementary Modulation Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 111: Up/down output trigger signal of master channel |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 1001: Count-up/-down mode |
| 0 | TAUDnMD0 | 0: INTTAUDnIm is not generated at the beginning of operation or at a restart time. |

#### (2) TAUDnCMURm for slave channels 2, 4, and 6

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.253 Contents of the TAUDnCMURm Register for Slave Channel 2, 4, and 6 of the Complementary Modulation Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Output mode for slave channels 2, 4, and 6

**Table 25.254 Control Bit Settings in Synchronous Channel Output Mode 2 with Complementary Modulation Output**

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 1: Synchronous channel output |
| TAUDnTOC.TAUDnTOCm | 1: Operating mode 2 |
| TAUDnTOL.TAUDnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUDnTDE.TAUDnTDEm | 1: Enables dead time operation. |
| TAUDnTDM.TAUDnTDMm | 0: Adds dead time if an interrupt is detected on an even upper channel and the conditions set by TAUDnTDL.TAUDnTDLm are satisfied. |
| TAUDnTDL.TAUDnTDLm | 0: Adds dead time to normal phase.<br>1: Adds dead time to reverse phase. |
| TAUDnTRE.TAUDnTREm | 1: Enables real-time output. |
| TAUDnTRO.TAUDnTROm | 0: Real-time output is low.<br>1: Real-time output is high. |
| TAUDnTRC.TAUDnTRCm | 0: Upper channel generates a real-time output trigger for channel m. |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation<br>1: Enables modulation |

**CAUTION**

At the PWM output, set TAUDnTDL.TAUDnTDLm exclusively from odd channels.

### (4) Simultaneous rewrite for slave channels 2, 4, and 6

Both the master and slave channels should have the same simultaneous rewrite settings.

**Table 25.255 Simultaneous Rewrite Settings for Slave Channels 2, 4, and 6 of Complementary Modulation Output Function**

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Monitors master channel for simultaneous rewrite triggers.<br>1: Monitors upper channel other than the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 1: A simultaneous rewrite trigger signal is generated when master channel starts to count and the corresponding slave channel is at the peak of a triangular wave. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. Monitors master channel for simultaneous rewrite triggers, regardless of the value of this bit, when TAUDnRDS.TAUDnRDSm = 0. |

### 25.16.3.7 Register settings for slave channels 3, 5, and 7

#### (1) TAUDnCMORm for slave channels 3, 5, and 7

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUDnCKS[1:0] | | TAUDnCCS[1:0] | | TAUDn MAS | TAUDnSTS[2:0] | | | TAUDnCOS[1:0] | | — | TAUDnMD[4:1] | | | | TAUDn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 25.256 Contents of the TAUDnCMORm Register for Slave Channel 3, 5, and 7 of the Complementary Modulation Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUDnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3<br>The value of the TAUDnCKS[1:0] bits of the master and slave channels must be identical. |
| 13, 12 | TAUDnCCS[1:0] | 00: Uses an operation clock as a count clock |
| 11 | TAUDnMAS | 0: Slave channel |
| 10 to 8 | TAUDnSTS[2:0] | 110: Dead time trigger |
| 7, 6 | TAUDnCOS[1:0] | 00: Unused. Set to 00. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUDnMD[4:1] | 0100: One-count mode |
| 0 | TAUDnMD0 | 1: Enables start trigger detection while counting. |

#### (2) TAUDnCMURm for slave channels 3, 5, and 7

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUDnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.257 Contents of the TAUDnCMURm Register for Slave Channel 3, 5, and 7 of the Complementary Modulation Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUDnTIS[1:0] | 00: Unused. Set to 00. |

### (3) Output mode for slave channels 3, 5, and 7

Table 25.258 Control Bit Settings in Synchronous Channel Output Mode 2 with Complementary Modulation Output

| Bit Name | Setting |
|---|---|
| TAUDnTOE.TAUDnTOEm | 1: Enables independent channel output mode |
| TAUDnTOM.TAUDnTOMm | 1: Synchronous channel output |
| TAUDnTOC.TAUDnTOCm | 1: Operating mode 2 |
| TAUDnTOL.TAUDnTOLm | 0: Positive logic<br>1: Negative logic |
| TAUDnTDE.TAUDnTDEm | 1: Enables dead time operation. |
| TAUDnTDM.TAUDnTDMm | 0: Adds dead time if an interrupt is detected on an even upper channel and the conditions set by TAUDnTDL.TAUDnTDLm are satisfied. |
| TAUDnTDL.TAUDnTDLm | 0: Adds dead time to normal phase.<br>1: Adds dead time to reverse phase. |
| TAUDnTRE.TAUDnTREm | 1: Enables real-time output. |
| TAUDnTRO.TAUDnTROm | 0: Real-time output is low.<br>1: Real-time output is high. |
| TAUDnTRC.TAUDnTRCm | 0: Upper channel generates a real-time trigger for channel m. |
| TAUDnTME.TAUDnTMEm | 0: Disables modulation<br>1: Enables modulation |

**CAUTION**

At the PWM output, set TAUDnTDL.TAUDnTDLm exclusively from even channels.

### (4) Simultaneous rewrite for slave channels 3, 5, and 7

Both the master and slave channels should have the same simultaneous rewrite settings.

Table 25.259 Simultaneous Rewrite Settings for Slave Channels 3, 5, and 7 of Complementary Modulation Output Function

| Bit Name | Setting |
|---|---|
| TAUDnRDE.TAUDnRDEm | 1: Enables simultaneous rewrite. |
| TAUDnRDS.TAUDnRDSm | 0: Monitors master channel for simultaneous rewrite triggers.<br>1: Monitors upper channel other than the channel group for simultaneous rewrite triggers. |
| TAUDnRDM.TAUDnRDMm | 1: Simultaneous rewrite trigger signal is generated when master channel counter is started and the corresponding slave channel is at the peak of triangular wave. |
| TAUDnRDC.TAUDnRDCm | 0: Does not operate as a simultaneous rewrite trigger generation channel. Monitors master channel for simultaneous rewrite triggers, regardless of the value of this bit, when TAUDnRDS.TAUDnRDSm = 0. |

### 25.16.3.8 Operating Procedure for Complementary Modulation Output Function

Table 25.260 Operating Procedure for Complementary Modulation Output Function (1/2)

| | Operation | TAUDn Status |
|---|---|---|
| Initial Channel Setting | Master channel: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.16.3.4, Register Settings for the Master Channel**.<br><br>Slave channel 1: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.16.3.5, Register Settings for Slave Channel 1**.<br><br>Slave channels 2, 4, and 6: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.16.3.6, Register settings for slave channels 2, 4, and 6**.<br><br>Slave channels 3, 5, and 7: Set TAUDnCMORm and TAUDnCMURm registers and the channel output mode as described in **Section 25.16.3.7, Register settings for slave channels 3, 5, and 7**.<br><br>Set the value of TAUDnCDRm register of every channel. Set a pulse cycle using TAUDnCDRm of master channel, and an interrupt count of master channel to be ignored using TAUDnCDRm of slave channel 1. Also set a duty width in TAUDnCDRm of slave channels 2, 4, and 6, and a dead time delay on slave channels 3, 5, and 7.<br><br>Set TAUDnTRC.TAUDnTRCm to 1 on slave channel 1. | Channel operation is stopped. |

**Table 25.260 Operating Procedure for Complementary Modulation Output Function (2/2)**

Restart Operation

| | Operation | TAUDn Status |
|---|---|---|
| Start Operation | Set TAUDnTS.TAUDnTSm of master and slave channels to 1 simultaneously.<br>TAUDnTS.TAUDnTSm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm of master and slave channels is set to 1 and the counter starts counting down. |
| During Operation | TAUDnCDRm, TAUDnTRO.TAUDnTROm, TAUDnTME.TAUDnTMEm, and TAUDnTDL.TAUDnTDLm can be changed at any time.<br>TAUDnCNTm and TAUDnRSF.TAUDnRSFm can be read at any time.<br><br>TAUDnRDT.TAUDnRDTm can be changed during operation. | TAUDnCDRm value of master channel and slave channels 2 to 7 is loaded into TAUDnCNTm to perform counting down. TAUDnCDRm value of slave channel 1 is loaded and the counter waits for a master channel interrupt. When the counter of master channel reaches $0000_H$:<br>• INTTAUDnIm is generated.<br>• TAUDnCDRm value is reloaded into TAUDnCNTm to continue counting down.<br>• TAUDnCNTm value of slave channel 1 decrements by 1 and the counter waits for the next master channel interrupt.<br>• TAUDnCNTm of slave channels 2, 4, and 6 reloads the TAUDnCDRm value, or performs counting in opposite direction.<br>• At the same timing when the TAUDnCDRm value of slave channels 2, 4, and 6 is loaded, the TAUDnTME.TAUDnTMEm value of slave channels 2 to 7 is reflected to the TAUDTTOUTm output.<br>• The counter of slave channel 1 waits for the next interrupt from the master channel when reaching $0000_H$. When the interrupt is detected:<br>– TAUDnCDRm value is reloaded into TAUDnCNTm and the counter waits for the next master channel interrupt.<br>– INTTAUDnIm is generated.<br>– TAUDnTRO.TAUDnTROm is changeable.<br>• When the counter of slave channels 2, 4, and 6 reaches $0001_H$:<br>– INTTAUDnIm is generated.<br>– PWM output of slave channel m is set/reset (when the specified condition of the channel output mode is matched).<br>– TAUDnCDRm value of slave channels 3, 5, and 7 is loaded into TAUDnCNTm to perform counting down.<br>• When the counter of slave channels 3, 5, and 7 reaches $0000_H$:<br>– INTTAUDnIm is generated.<br>– PWM output of slave channel m is set/reset (when the specified condition of the channel output mode is matched). |
| Stop Operation | Set TAUDnTT.TAUDnTTm of master and slave channels to 1 simultaneously.<br>TAUDnTT.TAUDnTTm is a trigger bit, which is automatically cleared to 0. | TAUDnTE.TAUDnTEm is cleared to 0 and the counter stops.<br>TAUDnCNTm and TAUDTTOUTm stop and retain their current values. |

### 25.16.3.9 Specific Timing Diagrams

The following settings apply to the timing diagram.

- Master channel: INTTAUDnIm is not generated at the beginning of operation. (TAUDnCMORm.TAUDnMD0 = 0)
- Slave channel 1: TAUDnCDRm = $0001_H$
- Slave channels 2 to 7: Positive logic (TAUDnTOL.TAUDnTOLm = 0)

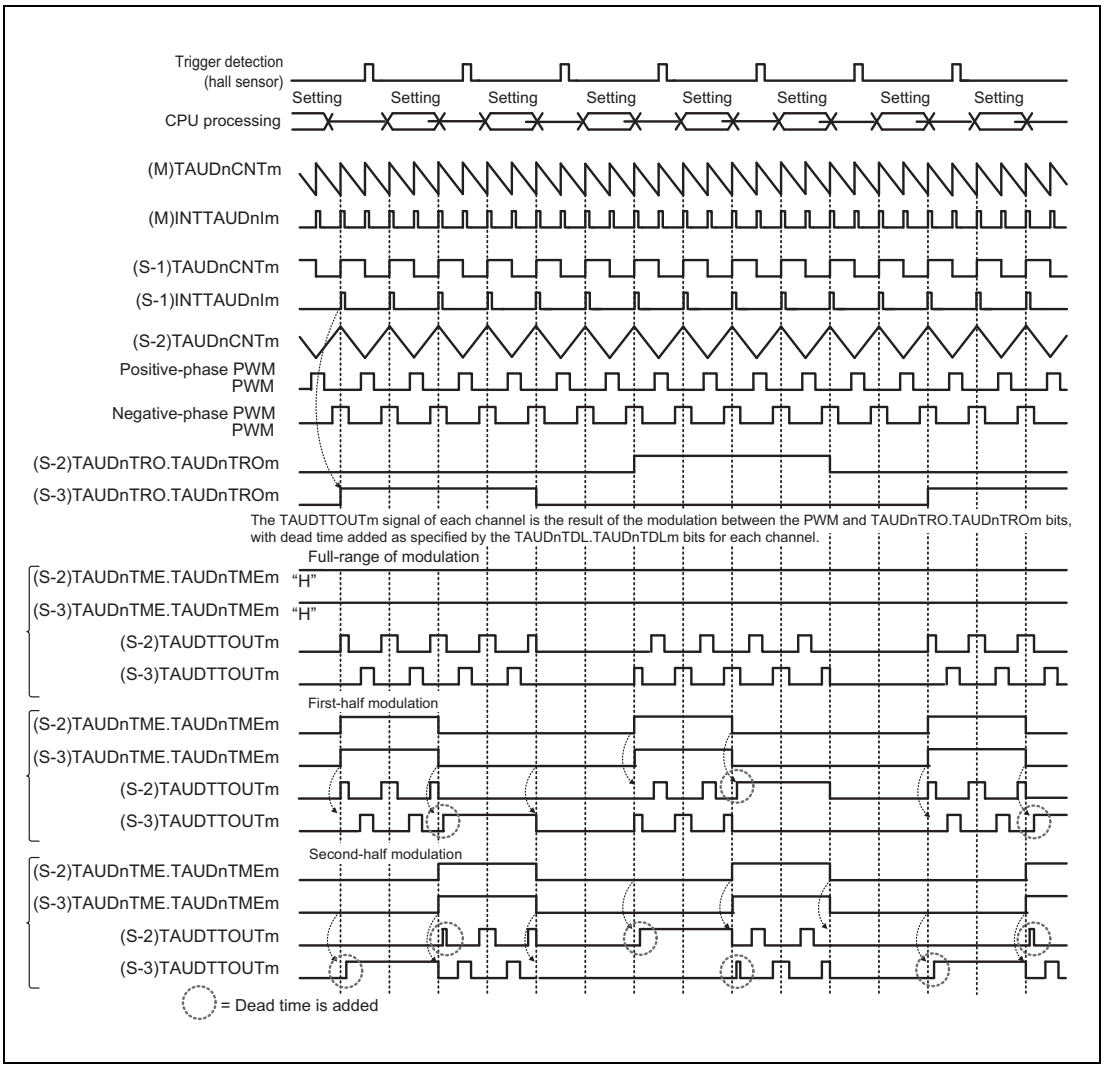



**Figure 25.137 Specific Timing Diagram of Complementary Modulation Output Function**

The above timing diagram shows how full modulation, first-half modulation, and second-half modulation can be achieved by modifying the TAUDnTME.TAUDnTMEm bits of lower slave channels during operation.

A modulated PWM output signal and TAUDnTRO.TAUDnTROm bit value are output from slave channels 2 and 3.

TAUDnTME.TAUDnTMEm and TAUDnTDL.TAUDnTDLm settings are reflected by detecting the count start timing and triangle PWM carrier cycle (peak interrupt timing).

TAUDnTRO.TAUDnTROm bit value is specified by software, but a new setting is applied only when an interrupt occurs on slave channel 1.

**NOTE**

Dead time is added to suppress simultaneous change of PWM edges of normal and reverse phases.

The “Setting”symbol indicates a time period when the values of TAUDnCDRm, TAUDnTME.TAUDnTMEm, TAUDnTRO.TAUDnTROm, and TAUDnTDL.TAUDnTDLm can be changed.

# Section 26 Timer Array Unit J (TAUJ)

This section contains a generic description of the timer array unit J (TAUJ).

The first part of this section describes the RH850/F1K specific features, such as the number of units and the register base addresses. The remainder of the section describes the functions and registers of the TAUJ.

## 26.1 Features of RH850/F1K TAUJ

### 26.1.1 Number of Units

This microcontroller has the following number of TAUJ units.

**Table 26.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of Units | 2 | 2 | 2 |
| Name | TAUJn (n = 0, 1) | TAUJn (n = 0, 1) | TAUJn (n = 0, 1) |

TAUJn has the following number of channels of timers.

**Table 26.2 TAUJn Unit Configurations and Channels**

| Unit Name (Channel Name) TAUJn | Number of Channels per Unit | RH850/F1K 100 pins (8 ch) | RH850/F1K 144 pins (8 ch) | RH850/F1K 176 pins (8 ch) |
|---|---|---|---|---|
| TAUJ0 | 4 | √ | √ | √ |
| TAUJ1 | 4 | √ | √ | √ |

**Table 26.3 Indices**

| Index | Description |
|---|---|
| n | Throughout this section, the individual TAUJ units are identified by the index “n”; for example, TAUJnTOM is the TAUJn channel output mode register. |
| m | The TAUJ has 4 channels. Throughout this section, the individual channels are identified by the index “m” (m = 0 to 3), thus a certain channel is denoted as CHm.<br>The even numbered channels (m = 0, 2) are denoted as CHm_even.<br>The odd numbered channels (m = 1, 3) are denoted as CHm_odd. |

### 26.1.2 Register Base Address

TAUJn base addresses are listed in the following table.

TAUJn register addresses are given as offsets from the base addresses.

**Table 26.4 Register Base Addresses**

| Base Address Name | Base Address |
|---|---|
| <TAUJ0_base> | $FFE5\ 0000_H$ |
| <TAUJ1_base> | $FFE5\ 1000_H$ |

### 26.1.3 Clock Supply

The TAUJn clock supply is shown in the following table.

**Table 26.5 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| TAUJ0 | PCLK | CKSCLK_ATAUJ | Timer count clock |
| | Register access clock | CPUCLK2 | Bus Clock |
| | | CKSCLK_ATAUJ | |
| TAUJ1 | PCLK | CKSCLK_IPERI1 | Timer count clock |
| | Register access clock | CPUCLK2 | Bus Clock |
| | | CKSCLK_IPERI1 | |

### 26.1.4 Interrupt Requests

TAUJn interrupt requests are listed in the following table.

**Table 26.6 Interrupt Requests**

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **TAUJ0** | | | |
| INTTAUJ0I0 | Channel 0 interrupt | 80 | 21 |
| INTTAUJ0I1 | Channel 1 interrupt | 81 | 80 |
| INTTAUJ0I2 | Channel 2 interrupt | 82 | 81 |
| INTTAUJ0I3 | Channel 3 interrupt | 83 | 22 |
| **TAUJ1** | | | |
| INTTAUJ1I0 | Channel 0 interrupt | 168 | 46 |
| INTTAUJ1I1 | Channel 1 interrupt | 169 | 100 |
| INTTAUJ1I2 | Channel 2 interrupt | 170 | 47 |
| INTTAUJ1I3 | Channel 3 interrupt | 171 | 101 |

### 26.1.5 Reset Sources

TAUJn reset sources are listed in the following table. TAUJn is initialized by these reset sources.

**Table 26.7 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| TAUJ0 | All reset sources except the transition to DeepSTOP mode (AWORES) |
| TAUJ1 | All reset sources (ISORES) |

### 26.1.6 External Input/Output Signals

External input/output signals of TAUJn are listed below.

**Table 26.8 External Input/Output Signals**

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **TAUJ0** | | |
| TAUJTTIN0, TAUJTTIN1 | Channel 0, 1 input | TAUJ0I0, TAUJ0I1 |
| TAUJTTIN2 | Channel 2 input | TAUJ0I2 or RTCA0OUT*1 |
| TAUJTTIN3 | Channel 3 input | TAUJ0I3 or RTCA0OUT*1 |
| TAUJTTOUT0 to TAUJTTOUT3 | Channel 0 to 3 output | TAUJ0O0 to TAUJ0O3 |
| **TAUJ1** | | |
| TAUJTTIN0 to TAUJTTIN3 | Channel 0 to 3 input | TAUJ1I0 to TAUJ1I3 |
| TAUJTTOUT0 to TAUJTTOUT3 | Channel 0 to 3 output | TAUJ1O0 to TAUJ1O3 |

Note 1. For details, see **Section 26.1.8, TAUJ0 Input Selection**.

### 26.1.7 Internal Input/Output Signals

The internal input/output signals of TAUJn are listed below.

**Table 26.9 Internal Input/Output Signals**

| Unit Signal Name | Description | Connected to |
|---|---|---|
| TAUJnTSSTm*1 | Simultaneous channel start trigger input | PIC |

Note 1. n = 1 only. TAUJ0TSSTm is not connected to PIC.

### 26.1.8 TAUJ0 Input Selection

The 1-Hz pulse output (RTCA0OUT) from RTCA0 and the output (TAUJTTOUT0) from TAUJ1 can be input to TAUJTTIN2 and TAUJTTIN3 as shown in the following figure.

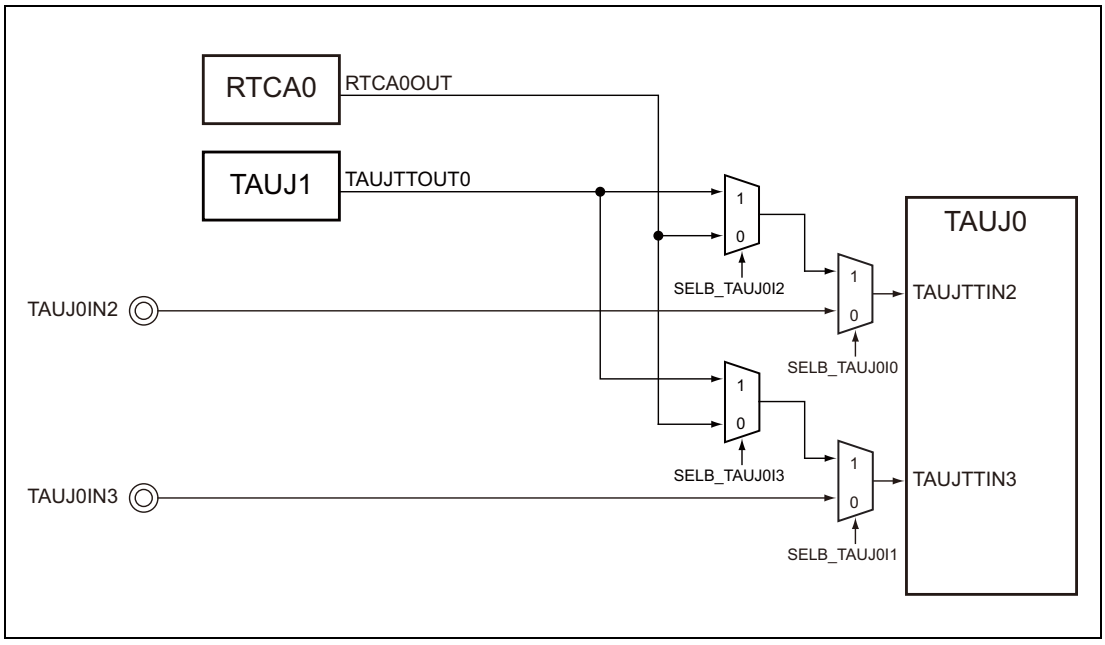



**Figure 26.1 Selection of Signals Input to TAUJ0**

The following table shows how to select signals input to the TAUJ.

**Table 26.10 TAUJ0 Input Selections**

| Input Signal | Function | Settings | | | |
|---|---|---|---|---|---|
| | | SELB_TAUJ0I3 | SELB_TAUJ0I2 | SELB_TAUJ0I1 | SELB_TAUJ0I0 |
| TAUJTTIN2 | Port TAUJ0I2 | — | — | — | 0 |
| | RTCA0OUT (Real-time clock 1-Hz output) | — | 0 | — | 1 |
| | TAUJTTOUT0 (TAUJ1) | — | 1 | — | 1 |
| TAUJTTIN3 | Port TAUJ0I3 | — | — | 0 | — |
| | RTCA0OUT (Real-time clock 1-Hz output) | 0 | — | 1 | — |
| | TAUJTTOUT0 (TAUJ1) | 1 | — | 1 | — |

### 26.1.8.1 List of Registers

Input signal selection register is listed in the following table.

**Table 26.11 List of Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| SL_TAUJ0 | TAUJTTINm Input Signal Selection Register | SELB_TAUJ0I | FFE5 $4000_H$ |

### 26.1.8.2 SELB_TAUJ0I — TAUJTTINm Input Signal Selection Register

This register selects the TAUJ0 input signals.

**Access:** This register can be read or written in 8-bit units.

**Address:** FFE5 $4000_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | SELB_TAUJ0I3 | SELB_TAUJ0I2 | SELB_TAUJ0I1 | SELB_TAUJ0I0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 26.12 SELB_TAUJ0I Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned. When writing to these bits, write the value after reset. |
| 3 | SELB_TAUJ0I3 | Selection of TAUJTTIN3 input signal:<br>0: RTCA0OUT<br>1: TAUJTTOUT0 |
| 2 | SELB_TAUJ0I2 | Selection of TAUJTTIN2 input signal:<br>0: RTCA0OUT<br>1: TAUJTTOUT0 |
| 1 | SELB_TAUJ0I1 | Selection of TAUJTTIN3 input signal:<br>0: Port TAUJ0I3<br>1: Timer Input |
| 0 | SELB_TAUJ0I0 | Selection of TAUJTTIN2 input signal:<br>0: Port TAUJ0I2<br>1: Timer Input |

# 26.2 Overview

## 26.2.1 Functional Overview

The TAUJ has the following functions:

- Independent channel operation function (operated using a single channel)
- Synchronous channel operation function (operated using a master channel and multiple slave channels)

The TAUJ is used to perform various count or timer operations and to output a signal which depends on the result of the operation. It contains one prescaler block for count clock generation and 4 channels, each equipped with a 32-bit counter TAUJnCNTm and a 32-bit data register TAUJnCDRm to hold the count start value or compare value.

It also contains several control and status registers.

### Independent and synchronous operation

Every channel can operate in two operating modes, either independently or in combination with other channels (synchronously). When one master channel and one or more slave channels operate in combination, the slave channels depend on the master channel.

When a channel is operated independently, it can be operated independent of all other channels.

The synchronous operation function is implemented by using a combination of channel groups (comprised of master and slave channels).

Several rules apply to the settings of channels.

## 26.2.2 Terms

In this section, the following terms are used.

### Independent channel operation function/synchronous operation channel operation function

TAUJ has 4 channels, and provides an independent channel operation function that individual channels operate independently and a synchronous channel operation function that is implemented by using a combination of channels.

- The independent channel operation function can be used any channel independently of all other channels.
- The synchronous channel operation function is implemented by using a combination of channel groups (comprised of master and slave channels).

Several rules apply to the settings of channels.

### Channel group

In the synchronous channel operation function, all channels that depend on each other are referred to as a “channel group”.

A channel group has one master channel and one or more slave channels.

### Upper/lower channel

Based on the channel number m, a channel with a smaller channel number or higher channel number can be referred to as “upper” or “lower” channel:

- Upper channel: Channel with a smaller channel number
- Lower channel: Channel with a larger channel number

For instance, as to channel 2, channel 1 is an upper channel and channel 3 is a lower channel. Channel 0 is the highest channel and channel 3 is the lowest channel.

### 26.2.3 Functional List of Timer Operations

This timer provides the following functions by operating each channel independently or by combining multiple channels.

**Table 26.13 Functional List of TAUJ Operations**

| Operation Function | Example |
|---|---|
| **Independent Channel Operation Functions** | **Section 26.12** |
| Interval Timer Function | **Section 26.12.1** |
| TAUJTTINm Input Interval Timer Function | **Section 26.12.2** |
| TAUJTTINm Input Pulse Interval Measurement Function | **Section 26.12.3** |
| TAUJTTINm Input Signal Width Measurement Function | **Section 26.12.4** |
| TAUJTTINm Input Position Detection Function | **Section 26.12.5** |
| TAUJTTINm Input Period Count Detection Function | **Section 26.12.6** |
| Overflow Interrupt Output Function (during TAUJTTINm Width Measurement) | **Section 26.12.7** |
| Overflow Interrupt Output Function (during TAUJTTINm Input Period Count Detection) | **Section 26.12.8** |
| **Synchronous Channel Operation Functions** | **Section 26.13** |
| PWM Output Function | **Section 26.13.1** |

### 26.2.4 TAUJ I/O and Interrupt Request Signals

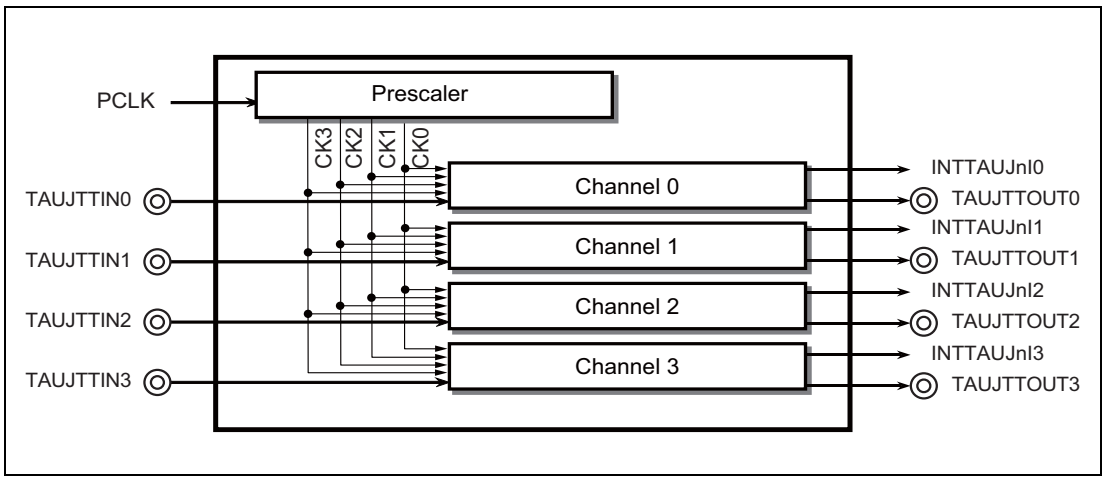



**Figure 26.2 TAUJ I/O and Interrupt Request Signals**

## 26.2.5 Block Diagram

The following figure shows the main components of the TAUJ.

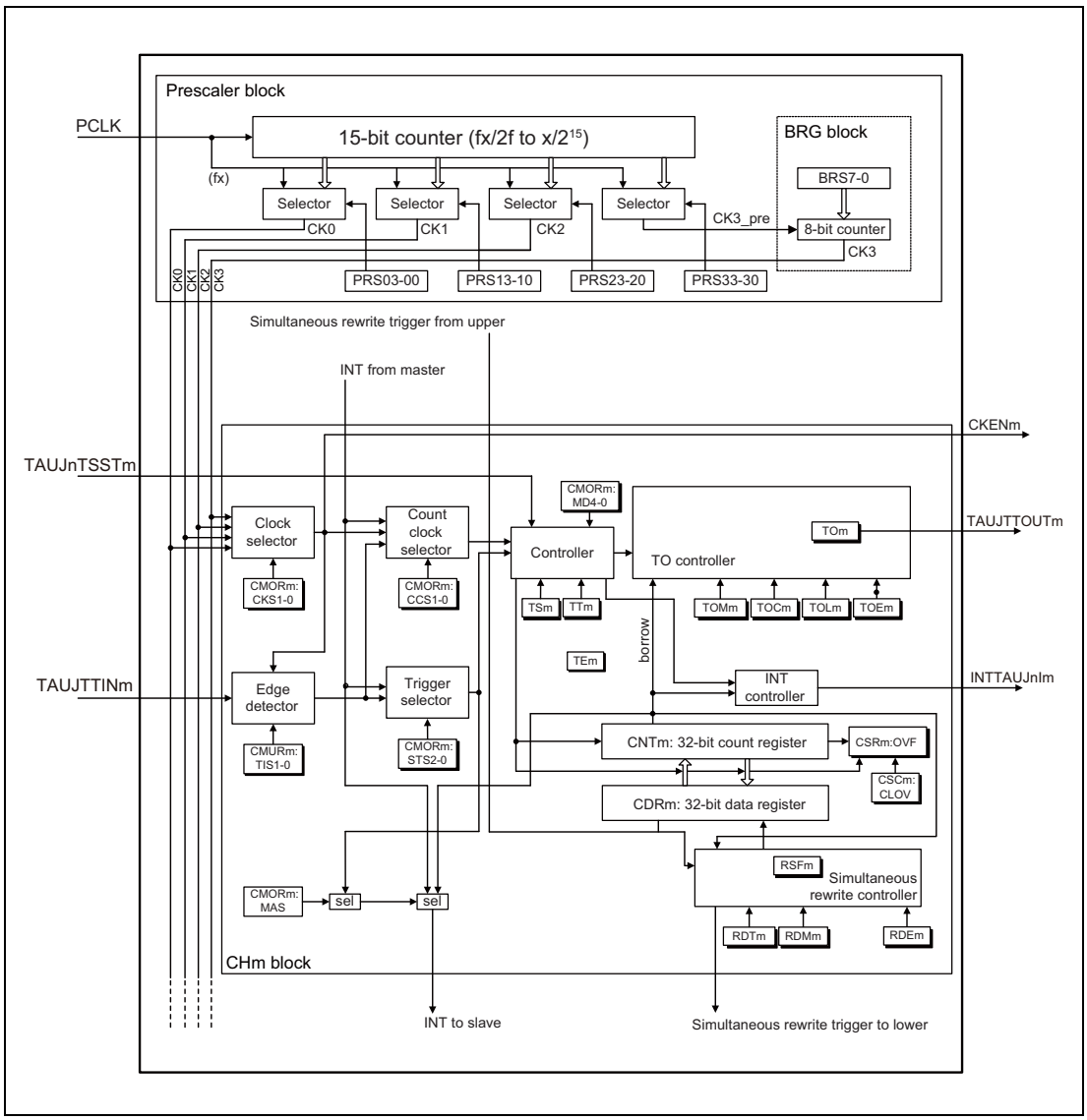



**Figure 26.3 Block Diagram of the TAUJ**

The prefix “TAUJn” has been omitted from the register names for the sake of clarity in the above figure.

## 26.2.6 Description of Block Diagram

The following describes the functional blocks.

### Prescaler block

The prescaler block provides up to 4 clock signals (CK0 to CK3) that can be used as count clocks for all channels.

Count clocks CK0 to CK2 are derived by dividing PCLK in the prescaler division factor of $2^0$ to $2^{15}$. The fourth count clock, CK3, is derived by dividing PCLK by a division factor that is not a power of 2 by using the baud rate generator.

### Clock and count clock selection

For every channel, the count clock selector selects which of the following is used as the clock source.

– One of CK0 to CK3 clocks (selected by the clock selector)

### Controller

The controller controls the main operations of the counter.

- Operating mode (selected with the TAUJnCMORm.TAUJnMD[4:0] bits)
- Counter start enable (TAUJnTS.TAUJnTSm) and counter stop (TAUJnTT.TAUJnTTm)

When counter start is enabled, status flag TAUJnTE.TAUJnTEm is set.

### Trigger selector

The counter starts automatically when it is enabled (TAUJnTE.TAUJnTEm = 1), or it waits for an external start trigger signal. Any of the following signals can be used as the start trigger.

– Synchronous channel start trigger input TAUJnTSSTm
– Valid edge of the TAUJTTINm input signal
– INTTAUJnIm from the master channel

### Simultaneous rewrite controller

Simultaneous rewrite control is enabled in synchronous operating modes. The data registers of all channels in a channel group (TAUJnCDRm) can be rewritten at any time. The simultaneous rewrite controller ensures that new data register values of all channels become effective at the same time.

### TAUJnTO controller

The output control of every channel enables the generation of various output signals such as PWM signals.

# 26.3 Registers

## 26.3.1 List of Registers

TAUJ registers are listed in the following table.

For details about <TAUJn_base>, see **Section 26.1.2, Register Base Address**.

**Table 26.14 List of Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| **TAUJn prescaler registers** | | | |
| TAUJn | TAUJn prescaler clock select register | TAUJnTPS | <TAUJn_base> + $90_H$ |
| | TAUJn prescaler baud rate setting register | TAUJnBRS | <TAUJn_base> + $94_H$ |
| **TAUJn control registers** | | | |
| TAUJn | TAUJn channel data register m | TAUJnCDRm | <TAUJn_base> + m × $4_H$ |
| | TAUJn channel counter register m | TAUJnCNTm | <TAUJn_base> + $10_H$ + m × $4_H$ |
| | TAUJn channel mode OS register m | TAUJnCMORm | <TAUJn_base> + $80_H$ + m × $4_H$ |
| | TAUJn channel mode user register m | TAUJnCMURm | <TAUJn_base> + $20_H$ + m × $4_H$ |
| | TAUJn channel status register m | TAUJnCSRm | <TAUJn_base> + $30_H$ + m × $4_H$ |
| | TAUJn channel status clear trigger register m | TAUJnCSCm | <TAUJn_base> + $40_H$ + m × $4_H$ |
| | TAUJn channel start trigger register | TAUJnTS | <TAUJn_base> + $54_H$ |
| | TAUJn channel enable status register | TAUJnTE | <TAUJn_base> + $50_H$ |
| | TAUJn channel stop trigger register | TAUJnTT | <TAUJn_base> + $58_H$ |
| **TAUJn output registers** | | | |
| TAUJn | TAUJn channel output enable register | TAUJnTOE | <TAUJn_base> + $60_H$ |
| | TAUJn channel output register | TAUJnTO | <TAUJn_base> + $5C_H$ |
| | TAUJn channel output mode register | TAUJnTOM | <TAUJn_base> + $98_H$ |
| | TAUJn channel output configuration register | TAUJnTOC | <TAUJn_base> + $9C_H$ |
| | TAUJn channel output active level register | TAUJnTOL | <TAUJn_base> + $64_H$ |
| **TAUJn reload data registers** | | | |
| TAUJn | TAUJn channel reload data enable register | TAUJnRDE | <TAUJn_base> + $A0_H$ |
| | TAUJn channel reload data mode register | TAUJnRDM | <TAUJn_base> + $A4_H$ |
| | TAUJn channel reload data trigger register | TAUJnRDT | <TAUJn_base> + $68_H$ |
| | TAUJn channel reload status register | TAUJnRSF | <TAUJn_base> + $6C_H$ |
| **TAUJn emulation register** | | | |
| TAUJn | TAUJn emulation register | TAUJnEMU | <TAUJn_base> + $A8_H$ |

## 26.3.2 Details of TAUJn Prescaler Registers

### 26.3.2.1 TAUJnTPS — TAUJn Prescaler Clock Select Register

This register specifies clocks CK0, CK1, CK2, and CK3_PRE for all channels of the PCLK prescalers. CK3 is generated by dividing CK3_PRE by the factor specified in TAUJnBRS.

**Access:** This register can be read or written in 16-bit units.

**Address:** <TAUJn_base> + $90_H$

**Value after reset:** $FFFF_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUJnPRS3[3:0] | | | | TAUJnPRS2[3:0] | | | | TAUJnPRS1[3:0] | | | | TAUJnPRS0[3:0] | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 26.15 TAUJnTPS Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 12 | TAUJnPRS3 [3:0] | Specifies a CK3_PRE clock.<br>The CK3_PRE clock is an input clock of the BRG unit which supplies CK3 operation clocks to all channels. |

| TAUJnPRS3[3:0] | CK3_PRE clock |
|---|---|
| $0000_B$ | PCLK/$2^0$ |
| $0001_B$ | PCLK/$2^1$ |
| $0010_B$ | PCLK/$2^2$ |
| $0011_B$ | PCLK/$2^3$ |
| $0100_B$ | PCLK/$2^4$ |
| $0101_B$ | PCLK/$2^5$ |
| $0110_B$ | PCLK/$2^6$ |
| $0111_B$ | PCLK/$2^7$ |
| $1000_B$ | PCLK/$2^8$ |
| $1001_B$ | PCLK/$2^9$ |
| $1010_B$ | PCLK/$2^{10}$ |
| $1011_B$ | PCLK/$2^{11}$ |
| $1100_B$ | PCLK/$2^{12}$ |
| $1101_B$ | PCLK/$2^{13}$ |
| $1110_B$ | PCLK/$2^{14}$ |
| $1111_B$ | PCLK/$2^{15}$ |

The above bits are rewritable only when all the counters using CK3 are stopped (TAUJnTE.TAUJnTEm = 0).

**Table 26.15 TAUJnTPS Register Contents (2/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 to 8 | TAUJnPRS2 [3:0] | Specifies a CK2 clock. |

| TAUJnPRS2[3:0] | CK2 clock |
|---|---|
| $0000_B$ | PCLK/$2^0$ |
| $0001_B$ | PCLK/$2^1$ |
| $0010_B$ | PCLK/$2^2$ |
| $0011_B$ | PCLK/$2^3$ |
| $0100_B$ | PCLK/$2^4$ |
| $0101_B$ | PCLK/$2^5$ |
| $0110_B$ | PCLK/$2^6$ |
| $0111_B$ | PCLK/$2^7$ |
| $1000_B$ | PCLK/$2^8$ |
| $1001_B$ | PCLK/$2^9$ |
| $1010_B$ | PCLK/$2^{10}$ |
| $1011_B$ | PCLK/$2^{11}$ |
| $1100_B$ | PCLK/$2^{12}$ |
| $1101_B$ | PCLK/$2^{13}$ |
| $1110_B$ | PCLK/$2^{14}$ |
| $1111_B$ | PCLK/$2^{15}$ |

The above bits are rewritable only when all the counters using CK2 are stopped (TAUJnTE.TAUJnTEm = 0).

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | TAUJnPRS1 [3:0] | Specifies a CK1 clock. |

| TAUJnPRS1[3:0] | CK1 clock |
|---|---|
| $0000_B$ | PCLK/$2^0$ |
| $0001_B$ | PCLK/$2^1$ |
| $0010_B$ | PCLK/$2^2$ |
| $0011_B$ | PCLK/$2^3$ |
| $0100_B$ | PCLK/$2^4$ |
| $0101_B$ | PCLK/$2^5$ |
| $0110_B$ | PCLK/$2^6$ |
| $0111_B$ | PCLK/$2^7$ |
| $1000_B$ | PCLK/$2^8$ |
| $1001_B$ | PCLK/$2^9$ |
| $1010_B$ | PCLK/$2^{10}$ |
| $1011_B$ | PCLK/$2^{11}$ |
| $1100_B$ | PCLK/$2^{12}$ |
| $1101_B$ | PCLK/$2^{13}$ |
| $1110_B$ | PCLK/$2^{14}$ |
| $1111_B$ | PCLK/$2^{15}$ |

The above bits are rewritable only when all the counters using CK1 are stopped (TAUJnTE.TAUJnTEm = 0).

**Table 26.15 TAUJnTPS Register Contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 3 to 0 | TAUJnPRS0 [3:0] | Specifies a CK0 clock. |

| TAUJnPRS0[3:0] | CK0 clock |
|---|---|
| $0000_B$ | PCLK/$2^0$ |
| $0001_B$ | PCLK/$2^1$ |
| $0010_B$ | PCLK/$2^2$ |
| $0011_B$ | PCLK/$2^3$ |
| $0100_B$ | PCLK/$2^4$ |
| $0101_B$ | PCLK/$2^5$ |
| $0110_B$ | PCLK/$2^6$ |
| $0111_B$ | PCLK/$2^7$ |
| $1000_B$ | PCLK/$2^8$ |
| $1001_B$ | PCLK/$2^9$ |
| $1010_B$ | PCLK/$2^{10}$ |
| $1011_B$ | PCLK/$2^{11}$ |
| $1100_B$ | PCLK/$2^{12}$ |
| $1101_B$ | PCLK/$2^{13}$ |
| $1110_B$ | PCLK/$2^{14}$ |
| $1111_B$ | PCLK/$2^{15}$ |

The above bits are rewritable only when all the counters using CK0 are stopped (TAUJnTE.TAUJnTEm = 0).

**NOTE**

TAUJn clock input PCLK is defined in the first part of this section, **Section 26.1.3, Clock Supply**.

### 26.3.2.2 TAUJnBRS — TAUJn Prescaler Baud Rate Setting Register

This register specifies the division factor of prescaler clock CK3.

CK3 is generated by dividing CK3_PRE by the factor specified in this register plus one. The PCLK prescaler for CK3_PRE is specified in TAUJnTPS. TAUJnPRS3[3:0].

**Access:** This register can be read or written in 8-bit units.

**Address:** <TAUJn_base> + $94_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TAUJnBRS[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 26.16 TAUJnBRS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | TAUJnBRS [7:0] | Specifies a CK3_PRE clock division factor for generating CK3. |

| TAUJnBRS[7:0] | CK3 clock |
|---|---|
| $0000\ 0000_B$ | CK3_PRE / 1 |
| $0000\ 0001_B$ | CK3_PRE / 2 |
| $0000\ 0010_B$ | CK3_PRE / 3 |
| $0000\ 0011_B$ | CK3_PRE / 4 |
| : | : |
| $1111\ 1110_B$ | CK3_PRE / 255 |
| $1111\ 1111_B$ | CK3_PRE / 256 |

## 26.3.3 Details of TAUJn Control Registers

### 26.3.3.1 TAUJnCDRm — TAUJn Channel Data Register

This register functions either as a compare register or as a capture register, depending on the operating mode specified in TAUJnCMORm.TAUJnMD[4:1].

**Access:** This register can be read or written in 32-bit units..
- When this register functions as a capture register, only reading is possible. Write operation is ignored.
- When this register functions as a compare register, reading and writing is possible.

**Address:** <TAUJn_base> + $0_H$ + m × $4_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUJnCDR[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUJnCDR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 26.17 TAUJnCDRm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TAUJnCDR [31:0] | Data register for capture/compare values |

### 26.3.3.2 TAUJnCNTm — TAUJn Channel Counter Register

This is a channel m counter register.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <TAUJn_base> + $10_H$ + m × $4_H$

**Value after reset:** FFFF $FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUJnCNT[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUJnCNT[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 26.18 TAUJnCNTm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TAUJnCNT [31:0] | 32-bit counter value |

The read value depends on a counter, an operating mode change, or TAUJnTS.TAUJnTSm/ TAUJnTT.TAUJnTTm bit value.

The initial counter read value depends on the operating mode and how the counter is stopped.

- By a reset
- By a counter stop trigger (TAUJnTT.TAUJnTTm = 1)

The following table lists the initial counter read values after the counter is stopped (TAUJnTE.TAUJnTEm = 0) and re-enabled (TAUJnTS.TAUJnTSm = 1).

The table also contains the counter read value one count after the counter is enabled (TAUJnTS.TAUJnTSm = 1) with the counter waiting for a start trigger.

**Table 26.19 TAUJnCNTm Read Values after Re-Enabling Counter**

| Mode Name | Count Method (Up/Down) | TAUJnCNTm | | |
|---|---|---|---|---|
| | | Start Value*1 | After Stop Trigger | After One Count |
| Interval timer mode | Count down | FFFF FFFF$_H$ | Stop value | — |
| Capture mode | Count up | 0000 0000$_H$ | Stop value | — |
| One-count mode | Count down | FFFF FFFF$_H$ | Stop value | Stop value |
| Capture and one-count mode | Count up | 0000 0000$_H$ | Stop value | Capture value + 1 (TAUJnCDRm) |
| Count capture mode | Count up | 0000 0000$_H$ | Stop value | — |
| Gate count mode | Count down | FFFF FFFF$_H$ | Stop value | Stop value |
| Capture and gate count mode | Count up | 0000 0000$_H$ | Stop value | Stop value |

Note 1. The value set for TAUJnCNTm when operating mode is changed after a reset is deasserted

### 26.3.3.3 TAUJnCMORm — TAUJn Channel Mode OS Register

This register controls channel m operation.

**Access:** This register can be read or written in 16-bit units.
Writable only when the counter is stopped (TAUJnTE.TAUJnTEm = 0).

**Address:** <TAUJn_base> + $80_H$ + m × $4_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUJnCKS[1:0] | | TAUJnCCS[1:0] | | TAUJn MAS | TAUJnSTS[2:0] | | | TAUJnCOS[1:0] | | — | TAUJnMD[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 26.20 TAUJnCMORm Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUJnCKS[1:0] | Selects an operation clock, which is used with the TAUJTTINm input edge detection circuit.<br>Setting of TAUJnCMORm.TAUJnCCS[1:0] bits also allows the operation clock to serve as the TAUJnCNTm count clock. |

| TAUJn CKS1 | TAUJn CKS0 | Selection of Operation Clock |
|---|---|---|
| 0 | 0 | CK0 |
| 0 | 1 | CK1 |
| 1 | 0 | CK2 |
| 1 | 1 | CK3 |

| Bit Position | Bit Name | Function |
|---|---|---|
| 13, 12 | TAUJnCCS[1:0] | Selects a count clock for TAUJnCNTm counter. |

| TAUJn CCS1 | TAUJn CCS0 | Selection of Operation Clock |
|---|---|---|
| 0 | 0 | Operation clock specified by TAUJnCMORm.TAUJnCKS[1:0]. |
| 0 | 1 | Setting prohibited |
| 1 | 0 | |
| 1 | 1 | |

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 | TAUJnMAS | Specifies whether the channel is a master or slave channel during synchronous channel operation.<br>0: Slave<br>1: Master<br>This bit setting is valid only for even channels (CHm_even). Odd channels (CHm-odd) are fixed to 0. |

**Table 26.20 TAUJnCMORm Register Contents (2/3)**

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>10 to 8</td><td>TAUJnSTS[2:0]</td><td>Selects an external start trigger.
<table>
<tr><th>TAUJn STS2</th><th>TAUJn STS1</th><th>TAUJn STS0</th><th>Selection of Operation Clock</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Software trigger</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Valid edge of TAUJTTINm input signal, which is specified by TAUJnCMURm.TAUJnTIS[1:0].</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Valid edge of TAUJTTINm input signal is used as a start trigger and the reverse edge as a stop trigger.</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Setting prohibited</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>INTTAUJnIm of master channel is used as a start trigger</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td rowspan="3">Setting prohibited</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table></td></tr>
<tr><td>7, 6</td><td>TAUJnCOS[1:0]</td><td>Specifies the timing for updating capture register TAUJnCDRm and overflow flag TAUJnCSRm.TAUJnOVF of channel m.<br>These bits are only valid if channel m is for capture function (capture mode and capture & one-count mode).
<table>
<tr><th>TAUJn COS1</th><th>TAUJn COS0</th><th>TAUJnCDRm</th><th>TAUJnCSRm.TAUJnOVF</th></tr>
<tr><td>0</td><td>0</td><td rowspan="2">Updated when valid edge of TAUJTTINm input is detected.</td><td>Updated (cleared or set) when valid edge of TAUJTTINm input is detected.<br>• Set TAUJnCSRm.TAUJnOVF if a counter overflow has occurred since the last valid edge was detected.<br>• Clear TAUJnCSRm.TAUJnOVF if no counter overflow has occurred since the last valid edge was detected.</td></tr>
<tr><td>0</td><td>1</td><td>Set when a counter overflow occurs and cleared by setting TAUJnCSCm.TAUJnCLOV to 1.</td></tr>
<tr><td>1</td><td>0</td><td rowspan="2">Updated when valid edge of TAUJTTINm input is detected and when a counter overflow occurs.<br>• Detection of valid edge of TAUJTTINm input: The counter value is written into TAUJnCDRm.<br>• Occurrence of overflow: FFFF FFFF$_H$ is loaded into TAUJnCDRm. Detection of the next valid edge of TAUJTTINm is ignored.</td><td>No setting</td></tr>
<tr><td>1</td><td>1</td><td>Set when a counter overflow occurs and cleared by setting TAUJnCSCm.TAUJnCLOV to 1.</td></tr>
</table></td></tr>
<tr><td>5</td><td>Reserved</td><td>When read, the value after reset is returned. When writing, write the value after reset.</td></tr>
</table>

**Table 26.20 TAUJnCMORm Register Contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 4 to 0 | TAUJnMD[4:0] | Specifies an operating mode. |

| TAUJn MD4 | TAUJn MD3 | TAUJn MD2 | TAUJn MD1 | TAUJn MD0 | Functional Description |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1/0 | Interval timer mode |
| 0 | 0 | 0 | 1 | 1/0 | Setting prohibited |
| 0 | 0 | 1 | 0 | 1/0 | Capture mode |
| 0 | 0 | 1 | 1 | 0 | Setting prohibited |
| 0 | 1 | 0 | 0 | 1/0 | One-count mode |
| 0 | 1 | 0 | 1 | 1/0 | Setting prohibited |
| 0 | 1 | 1 | 0 | 0 | Capture and one-count mode |
| 0 | 1 | 1 | 1 | 1/0 | Setting prohibited |
| 1 | 0 | 0 | 0 | 0 | Setting prohibited |
| 1 | 0 | 0 | 1 | 0 | Setting prohibited |
| 1 | 0 | 1 | 0 | 1/0 | Setting prohibited |
| 1 | 0 | 1 | 1 | 1/0 | Count capture mode |
| 1 | 1 | 0 | 0 | 0 | Gate count mode |
| 1 | 1 | 0 | 1 | 0 | Capture and gate count mode |

| Mode | Role of TAUJnMD0 Bit |
|---|---|
| Interval timer mode<br>Capture mode<br>Count capture mode | Specifies whether INTTAUJnIm is generated at the beginning of count operation (when a start trigger is entered) or not.<br>0: INTTAUJnIm is not generated.<br>1: INTTAUJnIm is generated. |
| One-count mode | Enables/disables start trigger detection during counting.<br>0: Disables detection.<br>1: Enables detection.<br>**CAUTION**<br>In one-count mode, INTTAUJnIm signal is not output at the beginning of count operation. |
| Capture and one-count mode<br>Gate count mode<br>Capture and gate count mode | This bit should be set to 0.<br>**CAUTION**<br>INTTAUJnIm signal is not output at the beginning of count operation. In addition, start trigger detected during counting is disabled. |

### 26.3.3.4 TAUJnCMURm — TAUJn Channel Mode User Register

This register specifies a type of valid edge detection used for TAUJTTINm input.

**Access:** This register can be read or written in 8-bit units.

**Address:** <TAUJn_base> + $20_H$ + m × $4_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUJnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 26.21 TAUJnCMURm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUJnTIS[1:0] | Specifies a valid edge of TAUJTTINm input signal. |

| TAUJn TIS1 | TAUJn TIS0 | Functional Description |
|---|---|---|
| 0 | 0 | Falling edge |
| 0 | 1 | Rising edge |
| 1 | 0 | Detection of falling and rising edges (selection of low width measurement)<br>Start trigger: Falling edge<br>Stop trigger (capture): Rising edge |
| 1 | 1 | Detection of falling and rising edges (selection of high width measurement)<br>Start trigger: Rising edge<br>Stop trigger (capture): Falling edge |

Edge detection of TAUJTTINm input signal is based on the operation clock selected by TAUJnCMORm.TAUJnCKS[1:0].

### 26.3.3.5 TAUJnCSRm — TAUJn Channel Status Register

This register indicates the overflow status of channel m.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <TAUJn_base> + $30_H$ + m × $4_H$

**Value after reset:** $0x_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | TAUJnOVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | — | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 26.22 TAUJnCSRm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. |
| 1 | Reserved | When read, unknown value is returned. |
| 0 | TAUJnOVF | Indicates the counter overflow status:<br>0: No overflow occurs<br>1: Overflow occurs<br>This bit is used only in the following modes:<br>• Capture mode<br>• Capture and one-count mode<br><br>The function of this bit depends on the setting of control bits TAUJnCMORm.TAUJnCOS[1:0]. |

### 26.3.3.6 TAUJnCSCm — TAUJn Channel Status Clear Trigger Register

This register is a trigger register for clearing the overflow flag TAUJnCSRm.TAUJnOVF of channel m.

**Access:** This register is a write-only register that can be written in 8-bit units. It is always read as $00_H$.

**Address:** <TAUJn_base> + $40_H$ + m × $4_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | TAUJnCLOV |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 26.23 TAUJnCSCm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | TAUJnCLOV | 0: No function<br>1: Clears the overflow flag TAUJnCSRm.TAUJnOVF |

### 26.3.3.7 TAUJnTS — TAUJn Channel Start Trigger Register

This register enables the counter operation for each channel.

**Access:** This register is a write-only register that can be written in 8-bit units. It is always read as $00_H$.

**Address:** <TAUJn_base> + $54_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TAUJnTS03 | TAUJnTS02 | TAUJnTS01 | TAUJnTS00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | W | W | W | W |

**Table 26.24 TAUJnTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When writing, write the value after reset. |
| 3 to 0 | TAUJnTSm | Enables the counter operation for channel m:<br>0: No function<br>1: Enables the counter operation and sets TAUJnTE.TAUJnTEm = 1.<br><br>Only the counter operation is enabled even if TAUJnTE.TAUJnTEm = 1.<br>Whether the counter is started or not depends on the selected operating mode. |

### 26.3.3.8 TAUJnTE — TAUJn Channel Enable Status Register

This register indicates whether a counter operation is enabled.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <TAUJn_base> + $50_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TAUJnTE03 | TAUJnTE02 | TAUJnTE01 | TAUJnTE00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 26.25 TAUJnTE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned. |
| 3 to 0 | TAUJnTEm | Indicates whether channel m's counter operation is enabled.<br>0: Counter operation is disabled<br>1: Counter operation is enabled<br><br>This bit is set to 1 when trigger input of TAUJnTSSTm (synchronous channel start trigger signal) is detected or when TAUJnTS.TAUJnTSm is set to 1.<br>This bit is reset to 0 when TAUJnTT.TAUJnTTm is set to 1. |

### 26.3.3.9 TAUJnTT — TAUJn Channel Stop Trigger Register

This register stops the counter operation of each channel.

**Access:** This register is a write-only register that can be written in 8-bit units. It is always read as $00_H$.

**Address:** <TAUJn_base> + $58_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TAUJnTT03 | TAUJnTT02 | TAUJnTT01 | TAUJnTT00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | W | W | W | W |

**Table 26.26 TAUJnTT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When writing, write the value after reset. |
| 3 to 0 | TAUJnTTm | Stops channel m's counter operation.<br>0: No function<br>1: Stops the counter operation and resets TAUJnTE.TAUJnTEm.<br><br>TAUJnCNTm, TAUJnTO.TAUJnTOm, and TAUJTTOUTm retain the values provided before the counter is stopped. |

## 26.3.4 Details of TAUJn Simultaneous Rewrite Register

### 26.3.4.1 TAUJnRDE — TAUJn Channel Reload Data Enable Register

This register enables and disables simultaneous rewrite of the data register TAUJnCDRm. It also enables and disables simultaneous rewrite of the data register TAUJnTOLm for the PWM output function.

**Access:** This register can be read or written in 8-bit units. It can only be written when TAUJnTE.TAUJnTEm = 0.

**Address:** <TAUJn_base> + $A0_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TAUJnRDE03 | TAUJnRDE02 | TAUJnRDE01 | TAUJnRDE00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 26.27 TAUJnRDE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 to 0 | TAUJnRDEm | Enables/disables simultaneous rewrite of the data register of channel m:<br>0: Disables simultaneous rewrite<br>1: Enabled simultaneous rewrite |

### 26.3.4.2 TAUJnRDM — TAUJn Channel Reload Data Mode Register

This register selects when the signal that controls simultaneous rewrite is generated.

**Access:** This register can be read or written in 8-bit units. It can only be written when TAUJnTE.TAUJnTEm = 0.

**Address:** <TAUJn_base> + $A4_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TAUJnRDM03 | TAUJnRDM02 | TAUJnRDM01 | TAUJnRDM00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 26.28 TAUJnRDM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 to 0 | TAUJnRDMm | Specifies when the signal that triggers simultaneous rewrite is generated:<br>0: When the master channel counter starts counting<br>1: No function<br><br>These bits only apply when TAUJnRDE.TAUJnRDEm = 1. |

### 26.3.4.3 TAUJnRDT — TAUJn Channel Reload Data Trigger Register

This register triggers the simultaneous rewrite enabling state.

**Access:** This register is a write-only register that can be written in 8-bit units. It is always read as $00_H$.

**Address:** <TAUJn_base> + $68_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TAUJnRDT03 | TAUJnRDT02 | TAUJnRDT01 | TAUJnRDT00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | W | W | W | W |

**Table 26.29 TAUJnRDT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When writing, write the value after reset. |
| 3 to 0 | TAUJnRDTm | Triggers the simultaneous rewrite enabling state.<br>0: No function<br>1: Simultaneous rewrite enabling state is triggered. The simultaneous rewrite enabling flag (TAUJnRSFm) is set to 1. The system waits for the simultaneous rewrite trigger.<br>These bits only apply when:<br>• TAUJnRDE.TAUJnRDEm = 1 |

### 26.3.4.4 TAUJnRSF — TAUJn Channel Reload Status Register

This flag register indicates the simultaneous rewrite status.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <TAUJn_base> + $6C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TAUJnRSF03 | TAUJnRSF02 | TAUJnRSF01 | TAUJnRSF00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 26.30 TAUJnRSF Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned. |
| 3 to 0 | TAUJnRSFm | Indicates the simultaneous rewrite status.<br>0: Indicates that simultaneous rewrite has been completed due to the generation of simultaneous rewrite trigger.<br>1: Indicates that the system waits for a simultaneous rewrite trigger in the simultaneous rewrite enabling state (TAUJnRDFm = 1). |

## 26.3.5 Details of TAUJn Output Registers

### 26.3.5.1 TAUJnTOE — TAUJn Channel Output Enable Register

This register enables and disables independent channel output mode controlled by software.

**Access:** This register can be read or written in 8-bit units.

**Address:** <TAUJn_base> + $60_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TAUJnTOE03 | TAUJnTOE02 | TAUJnTOE01 | TAUJnTOE00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 26.31 TAUJnTOE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 to 0 | TAUJnTOEm | Enables/disables independent channel output function:<br>0: Disables independent timer output function (controlled by software)<br>1: Enables independent timer output function |

### 26.3.5.2 TAUJnTO — TAUJn Channel Output Register

This register specifies and reads the level of TAUJTTOUTm.

**Access:** This register can be read or written in 8-bit units.

**Address:** <TAUJn_base> + $5C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TAUJnTO03 | TAUJnTO02 | TAUJnTO01 | TAUJnTO00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 26.32 TAUJnTO Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 to 0 | TAUJnTOm | Specifies and reads the level of TAUJTTOUTm:<br>0: Low<br>1: High<br><br>Only TAUJnTOm bits for which Independent Channel Output function is disabled (TAUJnTOEm = 0) can be written. |

### 26.3.5.3 TAUJnTOM — TAUJn Channel Output Mode Register

This register specifies the output mode of each channel.

**Access:** This register can be read or written in 8-bit units. It can only be written when the counter is stopped (TAUJnTE.TAUJnTEm = 0).

**Address:** <TAUJn_base> + $98_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TAUJnTOM03 | TAUJnTOM02 | TAUJnTOM01 | TAUJnTOM00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 26.33 TAUJnTOM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 to 0 | TAUJnTOMm | Specifies the channel output mode:<br>0: Independent channel output mode<br>1: Synchronous channel output mode<br><br>The output mode depends on the settings of channel output control (TAUJnTOE.TAUJnTOEm) bits. |

### 26.3.5.4 TAUJnTOC — TAUJn Channel Output Configuration Register

This register specifies the output mode of each channel in combination with TAUJnTOMm.

**Access:** This register can be read or written in 8-bit units. It can only be written when the counter is stopped (TAUJnTE.TAUJnTEm = 0).

**Address:** <TAUJn_base> + $9C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TAUJnTOC03 | TAUJnTOC02 | TAUJnTOC01 | TAUJnTOC00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 26.34 TAUJnTOC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 to 0 | TAUJnTOCm | Specifies the output mode:<br>0: Operation mode 1 (= Toggle mode)<br>1: No function<br><br>This bit must be set to 0 for all output modes except independent channel output mode controlled by software.<br>The output mode also depends on TAUJnTOM.TAUJnTOMm, as shown in the following table. |

| TAUJn TOMm | TAUJn TOCm | Functional Description |
|---|---|---|
| 0 | 0 | Toggle mode: Toggling proceeds when INTTAUJnIm occurs. |
| 0 | 1 | No function |
| 1 | 0 | Synchronous channel operation mode 1: Set when INT occurs on the master channel and reset when INT occurs on the slave channel. |
| 1 | 1 | No function |

### 26.3.5.5 TAUJnTOL — TAUJn Channel Output Active Level Register

This register specifies the output logic of the channel output bit (TAUJnTO.TAUJnTOm).

**Access:** This register can be read or written in 8-bit units.

**Address:** <TAUJn_base> + $64_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TAUJnTOL03 | TAUJnTOL02 | TAUJnTOL01 | TAUJnTOL00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 26.35 TAUJnTOL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 to 0 | TAUJnTOLm | Specifies the output logic of the channel m output bit (TAUJnTO.TAUJnTOm):<br>0: Positive logic (active high)<br>1: Negative logic (active low)<br><br>These bits apply in all channel output modes except independent channel output mode controlled by software. |

## 26.3.6 TAUJn Emulation Register

### 26.3.6.1 TAUJnEMU — TAUJn Emulation Register

This register controls operation by SVSTOP.

**Access:** This register can be read or written in 8-bit units.
A write should be performed when counters are stopped (TAUJnTE.TAUJnTEm = 0) and (EPC.SVSTOP = 0).

**Address:** <TAUJn_base> + $A8_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TAUJn SVSDIS | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R |

**Table 26.36 TAUJnEMU Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | TAUJnSVSDIS | When EPC.SVSTOP bit = 0:<br>Supply of the count clock continues when the debugger takes control of the microcontroller (as in the breakpoint), regardless of the value of this bit (1 or 0).<br>When EPC.SVSTOP bit = 1:<br>0: The count clock is stopped when the debugger takes control of the microcontroller (as in the breakpoint).<br>1: Supply of the count clock continues when the debugger takes control of the microcontroller (as in the breakpoint). |
| 6 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

## 26.4 Operating Procedure

The following lists the general operation procedure for the TAUJn:

After a reset is deasserted, the operation of each channel is stopped. Clock supply is started and writing to each register is enabled. All circuits and registers of all channels are initialized. The control register of TAUJTTOUTm is also initialized and outputs a low level.

1. Set the TAUJnTPS and TAUJnBRS registers to specify the clock frequency of CK0 to CK3.
2. Configure the desired TAUJn function:
   - Set the operation mode
   - Set any other control bits
3. Enable the counter by setting the TAUJnTS.TAUJnTSm bit to 1.
   The counter starts to count immediately, or when an appropriate trigger is detected, depending on the bit settings.
4. If desired, and if possible for the configured function, stop the counter or perform a forced restart operation during count operation. The counter can be stopped by setting the TAUJnTT.TAUJnTTm bit to 1. The counter can be forcibly restarted by setting the TAUJnTS.TAUJnTSm bit to 1.
5. Stop the function by setting the TAUJnTT.TAUJnTTm bit to 1.

**NOTES**

1. A detailed description of the required control bits and the operation of the individual functions are given in **Section 26.12, Independent Channel Operation Functions** and **Section 26.13, Synchronous Channel Operation Functions**.
2. The function can be changed while the counter is stopped (TAUJnTE.TAUJnTEm = 0).

## 26.5 Concepts of Synchronous Channel Operation Function

The synchronous channel operation function is implemented by using a combination of channel groups (comprised of master and slave channels).

Several rules apply to the settings of channels.

These rules are detailed in **Section 26.5.1, Rules of Synchronous Channel Operation Function**.

The synchronous channel operation function are detailed in the following section.

- **Section 26.13, Synchronous Channel Operation Functions**

### 26.5.1 Rules of Synchronous Channel Operation Function

#### Number of master and slave channels

- Only even channels (CH0, CH2) can be set as master channels. Any channel other than CH0 can be set as a slave channel.
- Only channels lower than the master channel can be set as slave channels, and multiple slave channels can be set for one master channel.
  Example: If CH2 is a master channel, CH3 can be set as slave channel.
- If two master channels are used, slave channels cannot cross the master.
  Example: If CH0 and CH2 are master channels, CH1 can be set as slave channel for CH0, but CH3 cannot.

#### Operation clock

- The same operation clock should be set for the master channel and the synchronized slave channel. This is achieved by setting the same value in the TAUJnCMORm.TAUJnCKS[1:0] bits of the master and slave channels.

The basic concepts of master/slave usage and operation clocks are illustrated in **Figure 26.4**.

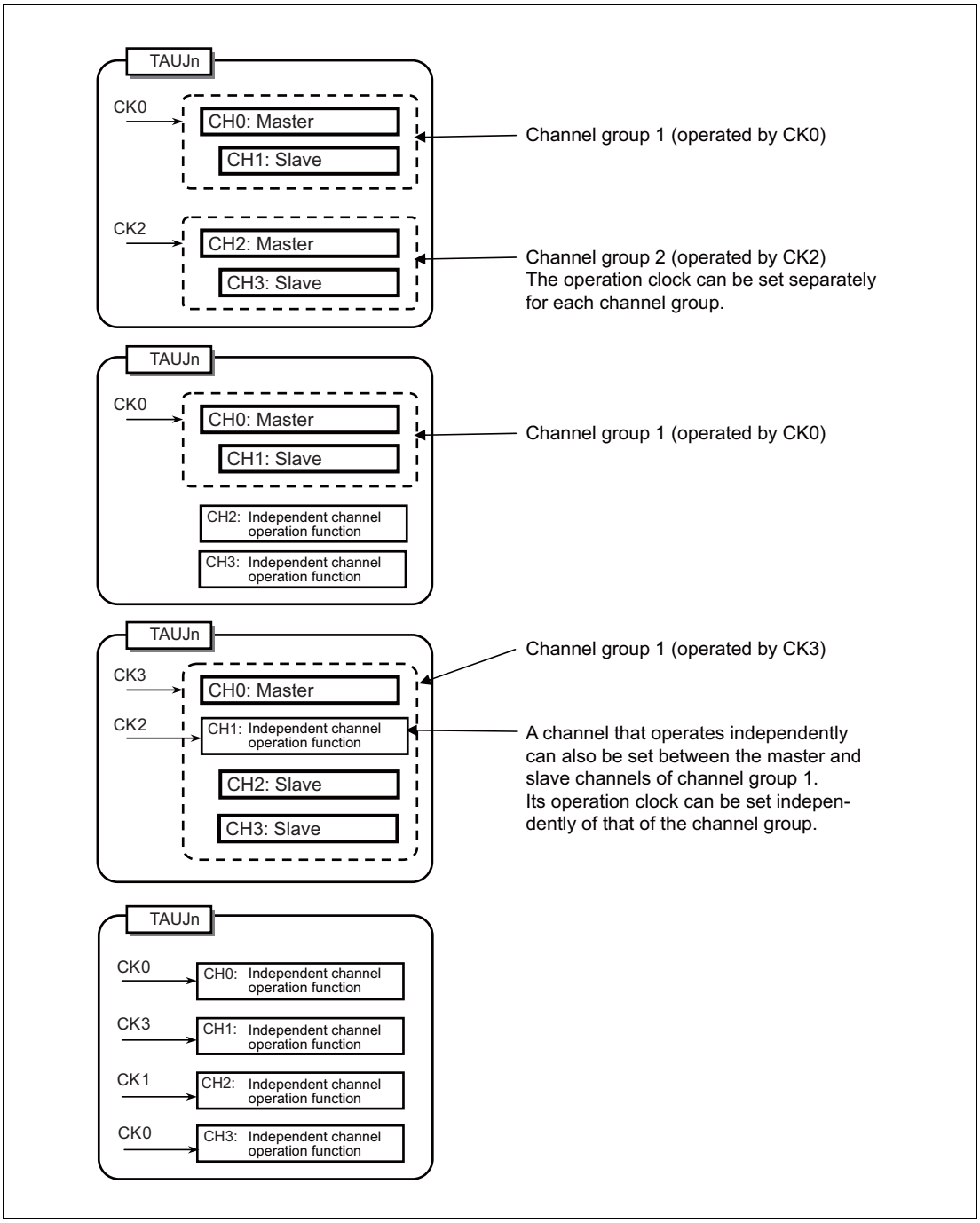



**Figure 26.4 Grouping of Channels and Assignment of Operation Clocks**

### 26.5.2 Simultaneous Start and Stop of Synchronous Channel Counters

Channels that are operated synchronously can be started and stopped simultaneously, both within a TAUJ unit and between TAUJ units.

#### 26.5.2.1 Simultaneous Start and Stop within a TAUJ Unit

- To simultaneously start synchronized channels, the TAUJnTS.TAUJnTSm bits of the channels should be set at the same time.
- To simultaneously stop synchronized channels, the TAUJnTT.TAUJnTTm bits of the channels should be set at the same time.

Setting 1 in the TAUJnTS.TAUJnTSm bits sets the corresponding TAUJnTE.TAUJnTEm bits to 1, enabling counting. The count start timing of the counter depends on the operating mode.

#### 26.5.2.2 Simultaneous Start between TAUJ Units

Counters in different TAUJ units can also be started simultaneously if the corresponding counters are enabled before receiving the simultaneous trigger signal.

# 26.6 Simultaneous Rewrite

## 26.6.1 How to Control Simultaneous Rewrite

The following figure shows the general procedure for simultaneous rewrite. The three main blocks (Initial settings, Start counter & count operation, and Simultaneous rewrite) are explained afterwards.

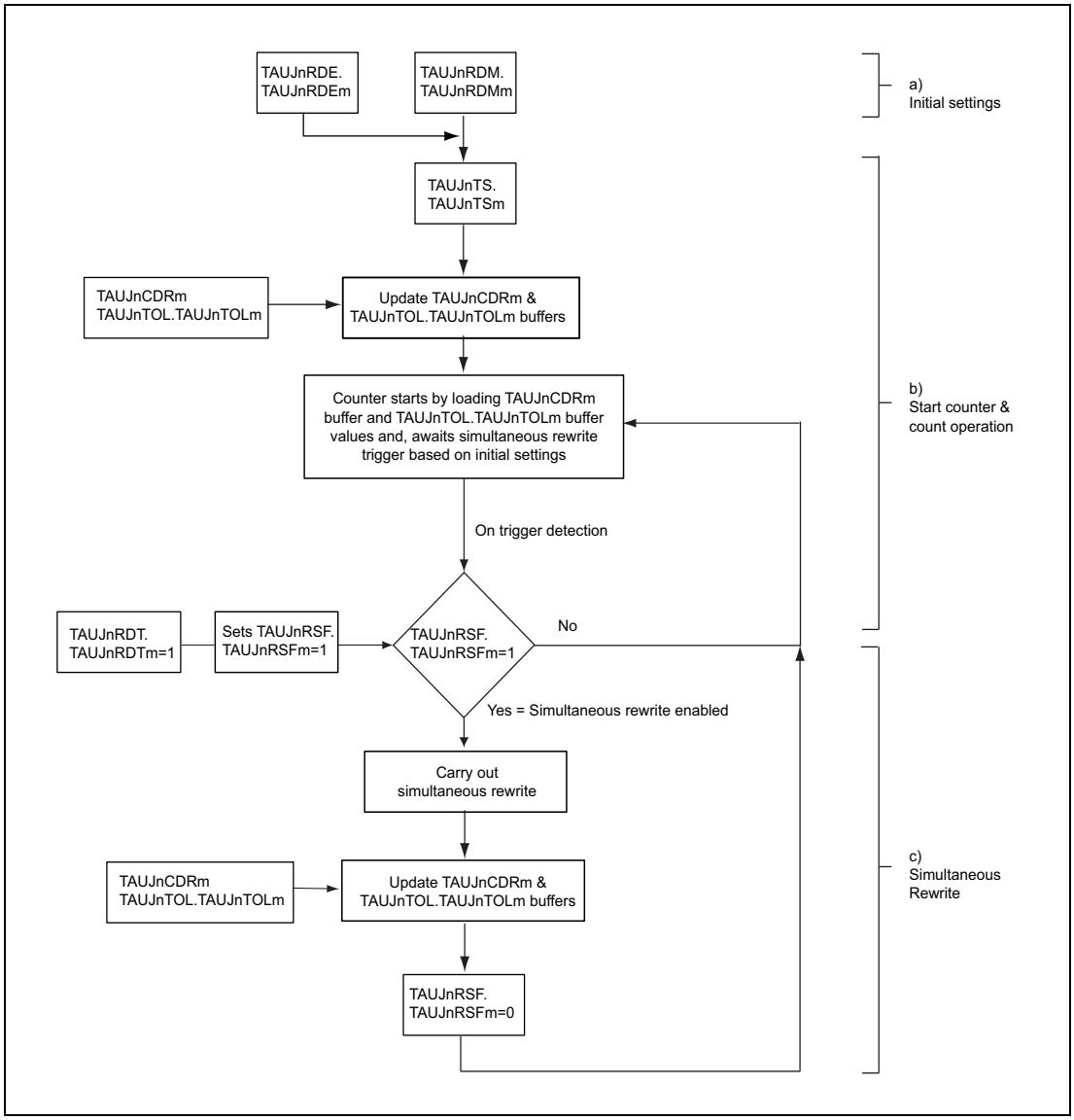



**Figure 26.5 General Procedure for Simultaneous Rewrite**

#### 26.6.1.1 Initial Settings

- To enable simultaneous rewrite in channel m, set TAUJnRDE.TAUJnRDEm = 1.
- To select simultaneous rewrite when the master channel starts counting, set TAUJnRDM.TAUJnRDMm.

#### 26.6.1.2 Start Counter and Count Operation

- To start all the TAUJnCNTm counters in the channel group, set the corresponding TAUJnTS.TAUJnTSm bits to 1. The values of TAUJnTOL.TAUJnTOLm and the data registers (TAUJnCDRm) are loaded into the corresponding TAUJnTOL.TAUJnTOLm buffer (TAUJnTOL.TAUJnTOLm buf) and data buffer registers (TAUJnCDRm buf) and the counters start.
- Setting the reload data trigger bit (TAUJnRDT.TAUJnRDTm) to 1 sets the reload flag (TAUJnRSF.TAUJnRSFm) to 1, enabling simultaneous rewrite. TAUJnRSF.TAUJnRSFm remains set to 1 until simultaneous rewrite is completed.
- When a specified trigger for simultaneous rewrite is detected, the TAUJnRSF.TAUJnRSFm bit is checked to see if simultaneous rewrite is enabled (TAUJnRSF.TAUJnRSFm = 1). If enabled, simultaneous rewrite is carried out. Otherwise simultaneous rewrite is not carried out and the system waits for detection of the next simultaneous rewrite trigger.

#### 26.6.1.3 Simultaneous Rewrite

- When the simultaneous rewrite trigger is detected and simultaneous rewrite is enabled (TAUJnRSF.TAUJnRSFm = 1), the current values of the data registers are copied to their buffers. These values are then loaded into the corresponding counters and the values are applied the next time the counter starts or restarts.
- When the simultaneous rewrite is complete, the TAUJnRSF.TAUJnRSFm bit is set to 0, and the system awaits the next simultaneous rewrite trigger.

### 26.6.2 Other General Rules for Simultaneous Rewrite

The following rules also apply.

- TAUJnRDE.TAUJnRDEm and TAUJnRDM.TAUJnRDMm cannot be changed while the counter is in operation (TAUJnTE.TAUJnTEm = 1).
- TAUJnTOL.TAUJnTOLm can be rewritten only during operation using the PWM output function. For all other functions, TAUJnTOL.TAUJnTOLm should be written before the counter starts. If it is rewritten while any other function is used, TAUJTTOUTm outputs an invalid waveform.

### 26.6.3 Simultaneous Rewrite Procedure

The simultaneous rewrite procedure with PWM output function is described in the following figure.

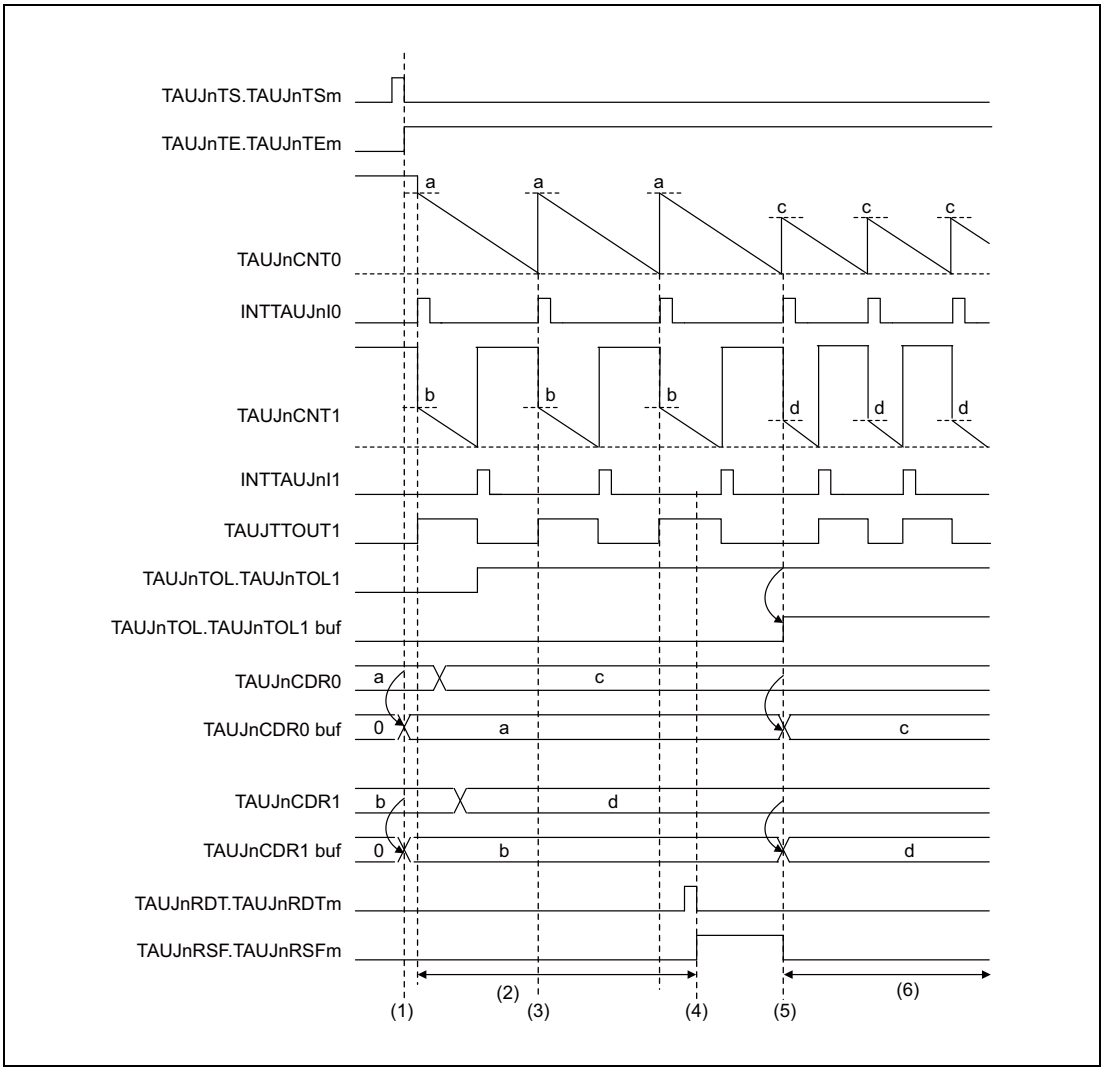



**Figure 26.6 Simultaneous Rewrite with PWM Output Function**

**Setting:**

CH0 is a master channel of PWM output function, and CH1 is a slave channel of PWM output function. Simultaneous rewrite is applied when the master channel starts counting.

**Description:**

(1) When TAUJnTS.TAUJnTSm = 1 is set, the value of TAUJnCDRm is copied to the TAUJnCDRm buffer and the value of TAUJnTOL.TAUJnTOLm is copied to the TAUJnTOL.TAUJnTOLm buffer.

(2) The TAUJnCDRm and TAUJnTOL.TAUJnTOLm registers can be written at any time.

(3) CH0 restarts counting, but simultaneous rewrite does not occur because it is disabled (TAUJnRSF.TAUJnRSFm = 0).

(4) The reload data trigger bit (TAUJnRDT.TAUJnRDTm) is set to 1 which sets the status flag (TAUJnRSF.TAUJnRSFm = 1), enabling simultaneous rewrite.

(5) Simultaneous rewrite is triggered when CH0 restarts counting, because simultaneous rewrite is enabled. The TAUJnCDRm value is loaded into the TAUJnCDRm buffer and the TAUJnTOL.TAUJnTOLm value is loaded into the TAUJnTOL.TAUJnTOLm buffer.

(6) The counters count down and await the next simultaneous rewrite trigger. The values of TAUJnCDRm and TAUJnTOL.TAUJnTOLm can be changed again.

## 26.7 Channel Output Modes

The output of the TAUJTTOUTm pin can be controlled in two ways, the latter of which can be further split into individual modes.

- By software (TAUJnTOE.TAUJnTOEm = 0)
  When controlled by software, the value written in the output register bit (TAUJnTO.TAUJnTOm) is sent to the output pin (TAUJTTOUTm).
- By TAUJ signals (TAUJnTOE.TAUJnTOEm = 1)
  When controlled by TAUJ signals, the output level of TAUJTTOUTm is set or reset or toggled by internal signals. The value of TAUJnTO.TAUJnTOm is updated accordingly to reflect the value of TAUJTTOUTm.
  - Independently (TAUJnTOM.TAUJnTOMm = 0)
    In case of independent operation, the output of the TAUJTTOUTm pin is only affected by settings of channel m. Therefore, independent channel operation should be selected (TAUJnTOM.TAUJnTOMm = 0).
  - Synchronously (TAUJnTOM.TAUJnTOMm = 1)
    In case of synchronous operation, the output of the TAUJTTOUTm pin is affected by settings of channel m and those of other channels. Therefore, synchronous channel operation should be selected for all synchronized channels (TAUJnTOM.TAUJnTOMm = 1).

The TAUJnTO.TAUJnTOm bit can always be read to determine the current value of TAUJTTOUTm, regardless of whether the pin is controlled by software, operated independently, or operated synchronously.

### Control bits

The settings of the control bits required to select a specific channel output mode are listed in **Table 26.37, Channel Output Modes**.

The channel output modes are described in details below.

- **Section 26.7.2, Channel Output Modes Controlled Independently by TAUJn Signals**
- **Section 26.7.3, Channel Output Modes Controlled Synchronously by TAUJn Signals**

### Batch operation of TAUJnTOm bit

Whether a set value is reflected to the TAUJnTOm bit or not is controlled by the TAUJnTOE.TAUJnTOEm bit.

The TAUJnTOm setting is written only to the bit (channel) set with TAUJnTOE.TAUJnTOEm bit = 0 when a write to the TAUJnTO register is attempted. No TAUJnTOm setting is reflected to the bit (channel) set with TAUJnTOE.TAUJnTOEm bit = 1.

**NOTE**

TAUJnTO.TAUJnTOm bit is placed so that its bit number corresponds to a channel number.

### Output logic

Positive logic or negative logic of the output is specified by control bit TAUJnTOL.TAUJnTOLm.

The value of TAUJnTOL.TAUJnTOLm bit should be set before the counter is started. It can only be changed during operation with PWM output function. Otherwise, changes to TAUJnTOL.TAUJnTOLm result in an undefined TAUJTTOUTm signal output.

See **Section 26.6, Simultaneous Rewrite**.

The various channel output modes and the channel output control bits are listed in **Table 26.37**.

**Table 26.37 Channel Output Modes**

| Channel Output Mode | TAUJnTOE.TAUJnTOEm | TAUJnTOM.TAUJnTOMm |
|---|---|---|
| **By software** | | |
| Independent channel output mode controlled by software | 0 | x |
| **By TAUJ signals, independently** | | |
| Independent channel output mode 1 | 1 | 0 |
| **By TAUJ signals, synchronously** | | |
| Synchronous channel output mode 1 | 1 | 1 |

- All combinations not listed in this table are forbidden.
- Bits marked with an x can be set to any value.

**NOTE**

The following bits cannot be changed during count operation (TAUJnTE.TAUJnTEm = 1):

- TAUJnTOM.TAUJnTOMm
- TAUJnTOC.TAUJnTOCm

### 26.7.1 General Procedures for Specifying a Channel Output Mode

This section describes the general procedures for specifying a TAUJTTOUTm channel output mode. The prerequisite is that timer output operation is disabled (TAUJnTOE.TAUJnTOEm = 0).

(1) Set TAUJnTO.TAUJnTOm to specify the initial level of the TAUJTTOUTm output.

(2) Set channel output mode according to **Table 26.37, Channel Output Modes**, and the output logic using the TAUJnTOL.TAUJnTOLm bit.

(3) Start the counter (TAUJnTS.TAUJnTSm = 1).

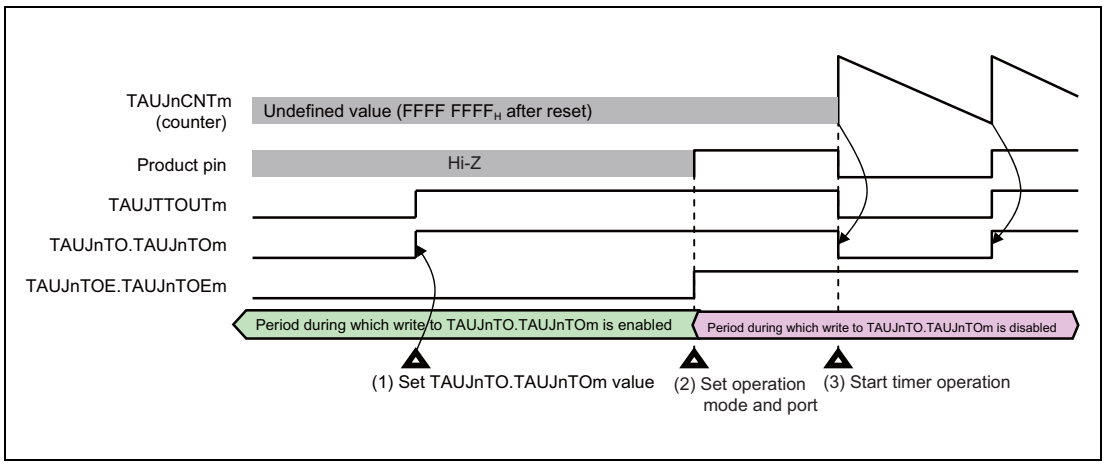



**Figure 26.7 General Procedure for Specifying a TAUJTTOUTm Channel Output Mode**

## 26.7.2 Channel Output Modes Controlled Independently by TAUJn Signals

This section lists the channel output modes that are controlled independently by TAUJn signals. The control bits used to specify a mode are listed in **Table 26.37, Channel Output Modes**.

### 26.7.2.1 Independent Channel Output Mode 1

#### Set/reset conditions

In this output mode, TAUJTTOUTm toggles when INTTAUJnIm is detected. The value of TAUJnTOL.TAUJnTOLm is ignored.

#### Prerequisites

There are no prerequisites other than those shown in **Table 26.37, Channel Output Modes**.

## 26.7.3 Channel Output Modes Controlled Synchronously by TAUJn Signals

This section lists the channel output modes that are controlled synchronously by TAUJn signals. The control bits used to specify a mode are listed in **Table 26.37, Channel Output Modes**.

### 26.7.3.1 Synchronous Channel Output Mode 1

#### Set/reset conditions

In this output mode, INTTAUJnIm of master channel serves as a set signal and INTTAUJnIm of the slave channel as a reset signal. If INTTAUJnIm of the master channel and INTTAUJnIm of the slave channel are generated at the same time, INTTAUJnIm of the slave channel (reset signal) has priority over INTTAUJnIm (set signal) of the master channel, i.e., the master channel is ignored.

#### Prerequisites

There are no prerequisites other than those shown in **Table 26.37, Channel Output Modes**.

## 26.8 Start Timing in Each Operating Modes

This section describes the timing at which the counter starts after TAUJnTS.TAUJnTSm is set to 1 in each operating mode.

In all modes, the value of data register and whether or not an interrupt occurs depends on mode and register settings.

**CAUTION**

**The count start timing described in this section is for your reference. Actually, the count start timing depends on the count clock timing.**

### 26.8.1 Interval Timer Mode, Capture Mode, and Count Capture Mode

The counter starts operating with the next count clock cycle after TAUJnTS.TAUJnTSm is set to 1. The value of data register is also loaded when the counter starts.

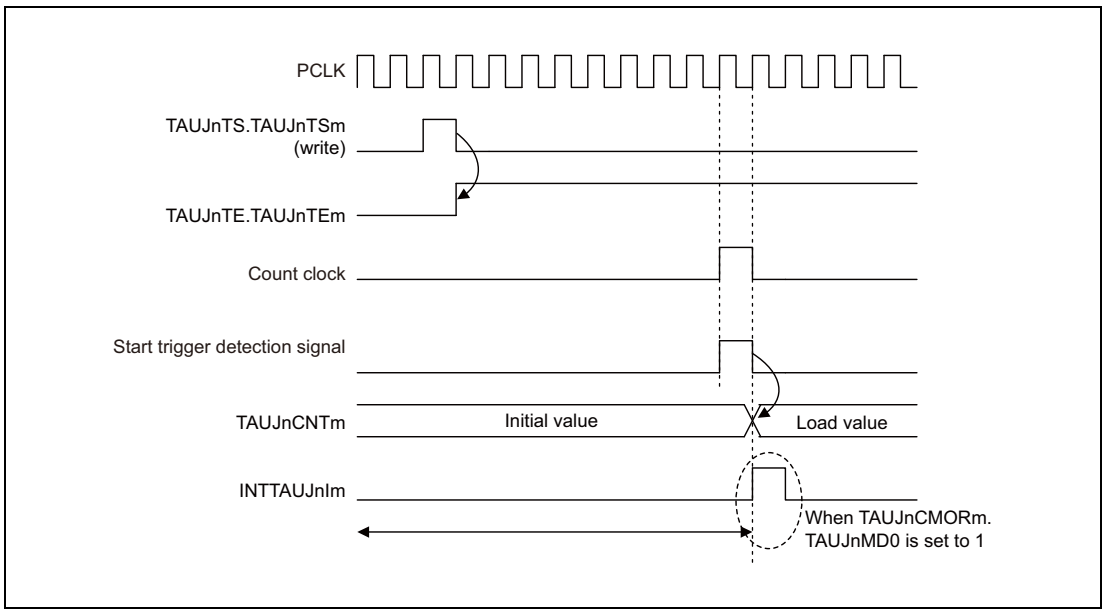



**Figure 26.8 Start Timing in Interval Timer Mode, Capture Mode, and Count Capture Mode**

### 26.8.2 Other Operating Modes

In other operating modes, count clock cycle is irrelevant to start of counter operation. The counter operation start timing is triggered only upon detection of a valid edge of TAUJTTINm. Once the counter starts, the value of data register is also loaded. The count clock cycles, which is irrelevant to start of counter operation, determine the frequency with which all operations take place.

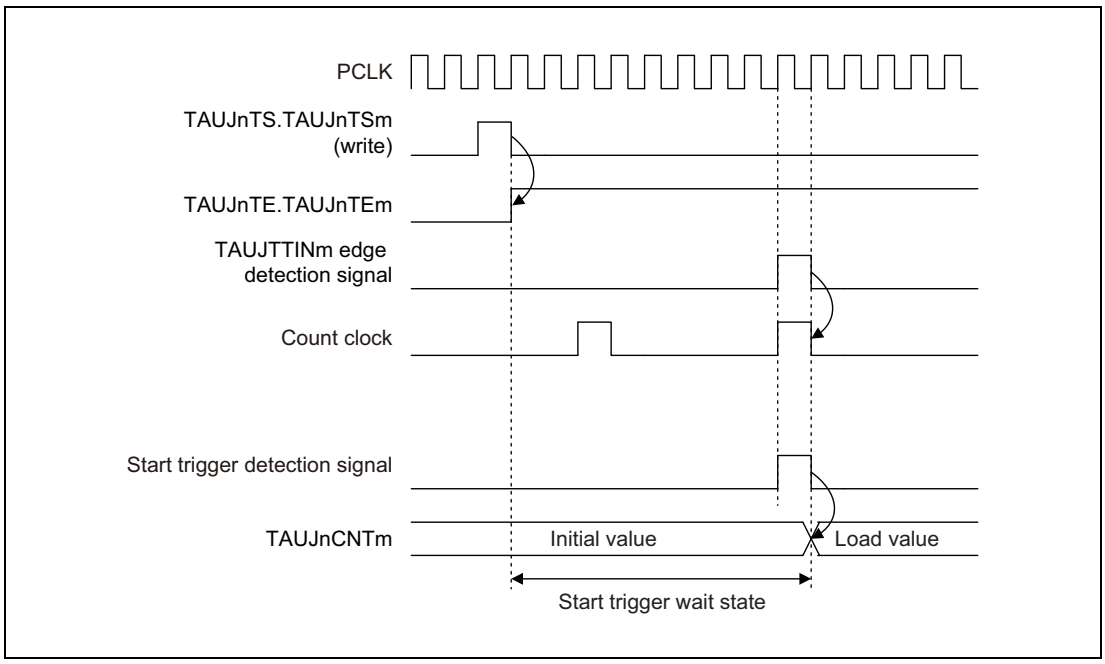



**Figure 26.9 Count Start Timing in Other Operating Modes**

## 26.9 TAUJTTOUTm Output and INTTAUJnIm Generation when Counter Starts or Restarts

When the counter starts, it is possible to specify whether an INTTAUJnIm is generated using the TAUJnCMORm.TAUJnMD0 bit. The generation of INTTAUJnIm when the TAUJnCMORm.TAUJnMD0 bit starts counting and the effect to TAUJTTOUTm depend on the selected function. For details, refer to the description of TAUJnCMORm.TAUJnMD0 of each function.

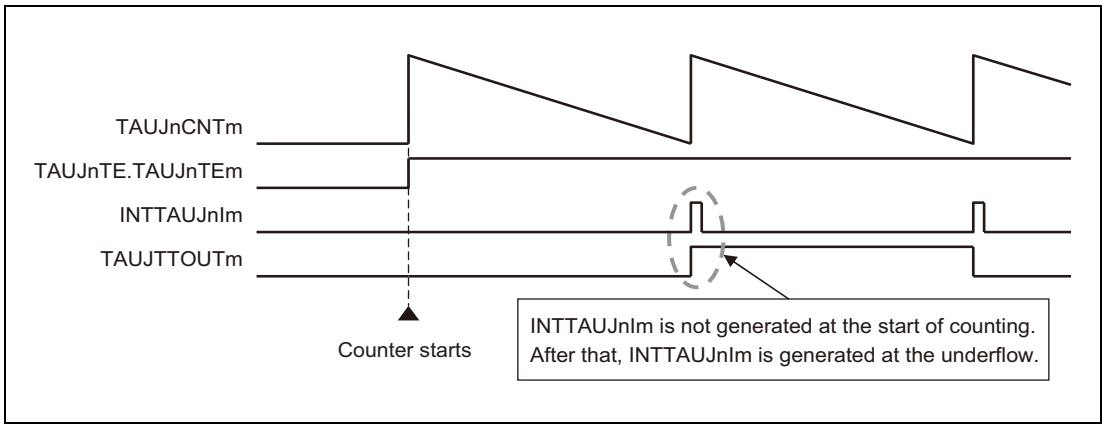



**Figure 26.10 INTTAUJnIm Generation Timing (when TAUJnCMORm.TAUJnMD0 = 0)**

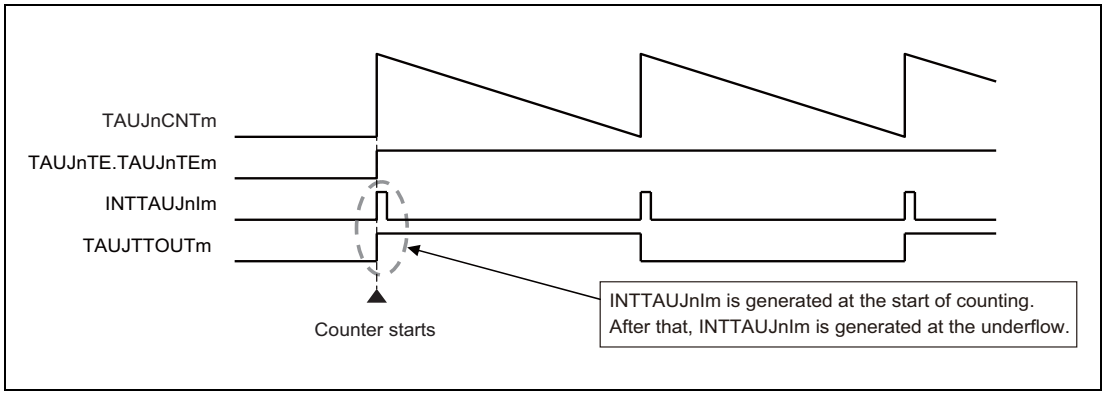



**Figure 26.11 INTTAUJnIm Generation Timing (when TAUJnCMORm.TAUJnMD0 = 1)**

## 26.10 Interrupt Generation upon Overflow

In certain independent functions, an interrupt is not generated when the counter value reaches FFFF $FFFF_H$ and an overflow occurs during count-up. This section describes how to generate an interrupt by combining channel operations in a mode that counts up and in a mode that counts down.

The appropriate operation mode for the second channel depends on the operation mode of the first channel. Nevertheless, the principle is the same for all combinations:

- Find an operation mode for the second channel that counts down in such a manner, that it reaches 0000 $0000_H$ at the same time as the first channel overflows (TAUJnCNTm = FFFF $FFFF_H$).
- Set TAUJnCDRm of the second channel to FFFF $FFFF_H$.
- The two channels must count at the same speed (i.e. they must have the same count clock).
- Both channels are triggered by the same TAUJTTINm input.
- The trigger detection settings (TAUJnCMORm.TAUJnSTS[2:0] and TAUJnCMURm.TAUJnTIS[1:0]) must be identical for both channels.

**Result:**

The down-counter of the second channel reaches 0000 $0000_H$ at exactly the same time as the up-counter of the first channel overflows (TAUJnCNTm = FFFF $FFFF_H$). Thus the second channel generates the desired interrupt.

The following sections list the operating modes that count down that are required to match specific operating modes that count up, as well as example timing diagrams.

### 26.10.1 Combination of the TAUJTTINm Input Position Detection Function and the Interval Timer Function

When the capture trigger is input simultaneously to TAUJTTINm of both channels, INTTAUJnIm of the interval timer function can detect the overflow when TAUJnCNTm of the TAUJTTINm input position detection function exceeds FFFF $FFFF_H$.

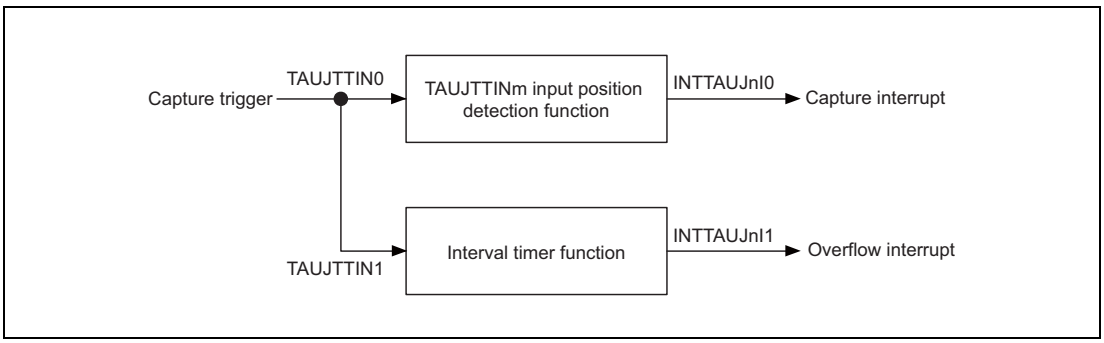



**Figure 26.12 Combination of the TAUJTTINm Input Position Detection Function and the Interval Timer Function**

**Timing diagram**

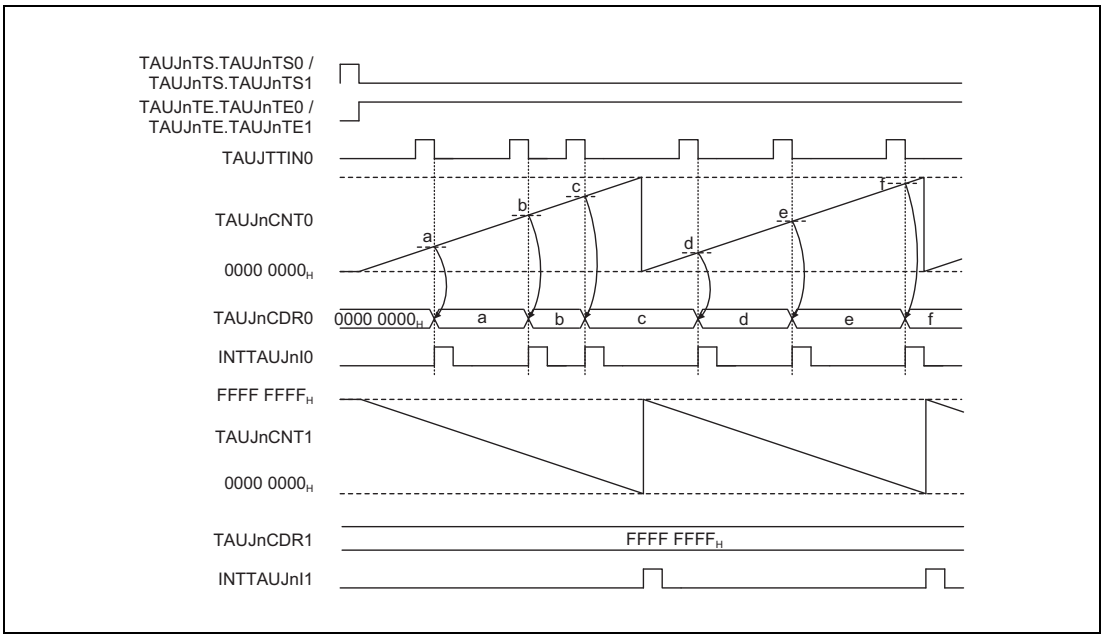



**Figure 26.13 Interrupt Generation via Combination of the TAUJTTINm Input Position Detection Function and the Interval Timer Function**

## 26.11 TAUJTTINm Edge Detection

Edge detection is based on the operation clock. This means that an edge can only be detected at the next rising edge of the operation clock. This can lead to a maximum delay of one operation clock cycle.

The following figure shows when edge detection takes place.

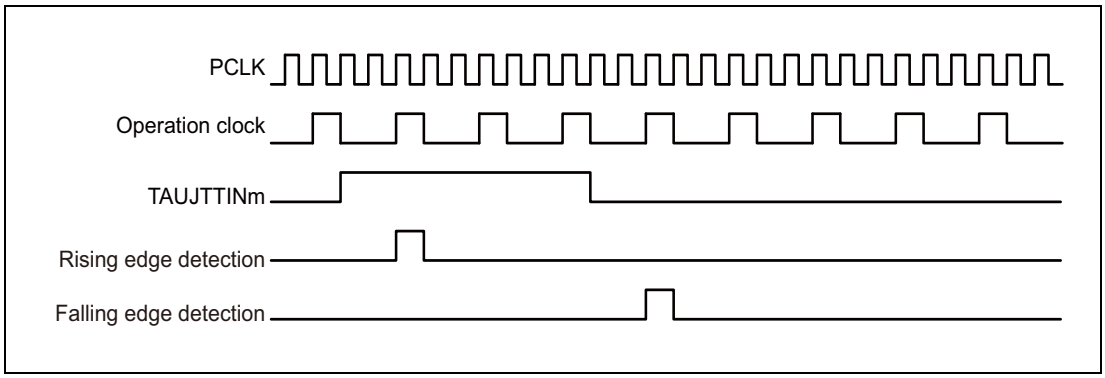



**Figure 26.14 Basic Edge Detection Timing**

**Figure 26.14** shows an image of operation timing. In the actual operation, delay time occurs due to noise filter and synchronization circuit between the TAUJnIm pin and TAUJn.

## 26.12 Independent Channel Operation Functions

The following sections list the independent channel operation functions provided by the TAUJ. For a general overview of independent channel operation functions, see **Section 26.2, Overview**.

### 26.12.1 Interval Timer Function

#### 26.12.1.1 Overview

**Summary**

This function is used as a reference timer for generating timer interrupts (INTTAUJnIm) at regular intervals. When an interrupt is generated, the TAUJTTOUTm signal toggles, resulting in a square wave.

**Functional description**

The counter is enabled by setting the channel trigger bit (TAUJnTS.TAUJnTSm) to 1. This in turn sets TAUJnTE.TAUJnTEm = 1, enabling count operation. The current value of TAUJnCDRm is loaded to TAUJnCNTm and the counter starts to count down from this value.

When the counter reaches 0000 $0000_H$, INTTAUJnIm is generated and the TAUJTTOUTm signal toggles. TAUJnCNTm then loads the TAUJnCDRm value and subsequently continues operation.

The value of TAUJnCDRm can be rewritten at any time, and the changed value of TAUJnCDRm is applied the next time the counter starts to count down.

The counter can be stopped by setting TAUJnTT.TAUJnTTm to 1, which in turn sets TAUJnTE.TAUJnTEm to 0. TAUJnCNTm and TAUJTTOUTm stop but retain their values. The counter can be restarted by setting TAUJnTS.TAUJnTSm to 1. The counter can also be forcibly restarted (without stopping it first) by setting TAUJnTS.TAUJnTSm to 1 during operation.

**Conditions**

If the TAUJnCMORm.TAUJnMD0 bit is set to 0, the first interrupt after a start or restart is not generated, and therefore TAUJTTOUTm does not toggle. This results in a reverted TAUJTTOUTm signal compared to when TAUJnCMORm.TAUJnMD0 is set to 1.

#### 26.12.1.2 Equations

INTTAUJnIm cycle = count clock cycle × (TAUJnCDRm + 1)

TAUJTTOUTm square wave cycle = count clock cycle × (TAUJnCDRm+1) × 2

### 26.12.1.3 Block Diagram and General Timing Diagram

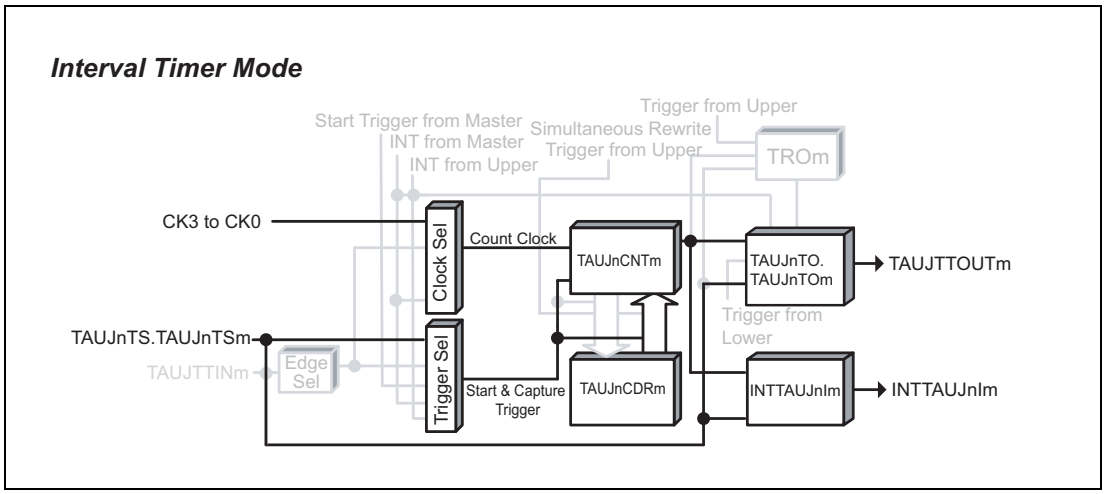



**Figure 26.15 Block Diagram for Interval Timer Function**

The following settings apply to the general timing diagram.

- INTTAUJnIm is generated at operation start (TAUJnCMORm.TAUJnMD0 = 1).

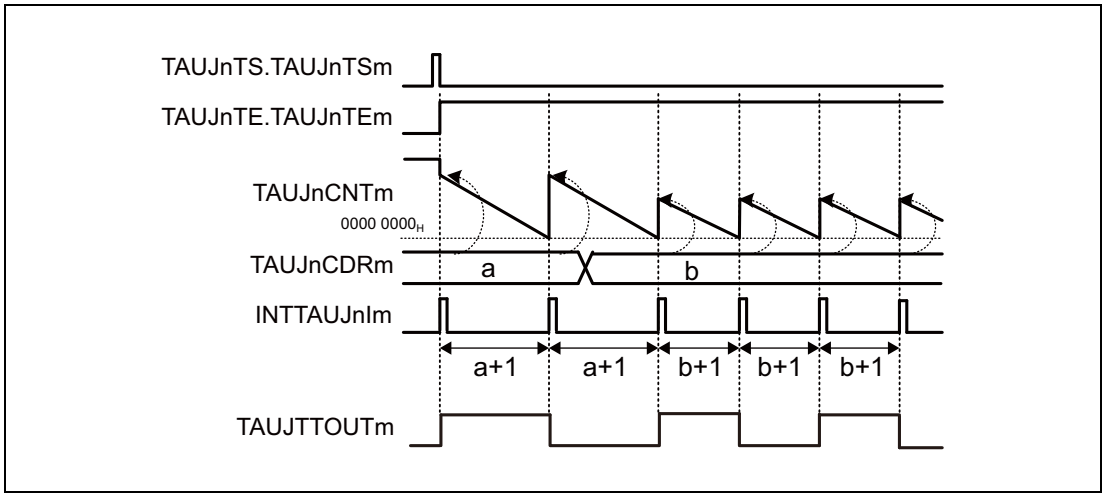



**Figure 26.16 General Timing Diagram for Interval Timer Function**

### 26.12.1.4 Register Settings

#### (1) TAUJnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUJnCKS[1:0] | | TAUJnCCS[1:0] | | TAUJn MAS | TAUJnSTS[2:0] | | | TAUJnCOS[1:0] | | — | TAUJnMD[4:1] | | | | TAUJn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 26.38 Contents of the TAUJnCMORm register for Interval Timer Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUJnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output CK0<br>01: Prescaler output CK1<br>10: Prescaler output CK2<br>11: Prescaler output CK3 |
| 13, 12 | TAUJnCCS[1:0] | Write $00_B$. |
| 11 | TAUJnMAS | Write $0_B$. |
| 10 to 8 | TAUJnSTS[2:0] | Write $000_B$. |
| 7, 6 | TAUJnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUJnMD[4:1] | Write $0000_B$. |
| 0 | TAUJnMD0 | 0: INTTAUJnIm is not generated and TAUJTTOUTm does not toggle when operation starts or restarts.<br>1: Generates INTTAUJnIm and toggles TAUJTTOUTm when operation starts or restarts. |

#### (2) TAUJnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUJnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 26.39 Contents of the TAUJnCMURm register for Interval Timer Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUJnTIS[1:0] | 00: Not used, so set to 00. |

### (3) Channel output mode

Table 26.40 Control Bit Settings in Independent Channel Output Mode 1

| Bit name | Setting |
|---|---|
| TAUJnTOE.TAUJnTOEm | Write $1_B$. |
| TAUJnTOM.TAUJnTOMm | Write $0_B$. |
| TAUJnTOC.TAUJnTOCm | Write $0_B$. |
| TAUJnTOL.TAUJnTOLm | Write $0_B$. |

**NOTE**

The channel output mode can also be set to Channel Output Mode Controlled by Software by setting TAUJnTOE.TAUJnTOEm = 0. TAUJTTOUTm can then be controlled independently of the interrupts. For details see **Section 26.7, Channel Output Modes**.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUJnRDE and TAUJnRDM) cannot be used with the interval timer function. Therefore, these registers must be set to 0.

Table 26.41 Simultaneous Rewrite Settings for Interval Timer Function

| Bit Name | Setting |
|---|---|
| TAUJnRDE.TAUJnRDEm | 0: Disables simultaneous rewrite. |
| TAUJnRDM.TAUJnRDMm | 0: When simultaneous rewrite is disabled (TAUJnRDE.TAUJnRDEm = 0), set these bits to 0. |

### 26.12.1.5 Operating Procedure for Interval Timer Function

Table 26.42 Operating Procedure for Interval Timer Function

| | Operation | Status of TAUJn |
|---|---|---|
| Initial channel setting | Set the TAUJnCMORm and TAUJnCMURm registers as described in **Table 26.38, Contents of the TAUJnCMORm register for Interval Timer Function** and **Table 26.39, Contents of the TAUJnCMURm register for Interval Timer Function**.<br><br>Set the value of the TAUJnCDRm register.<br><br>Set the channel output mode by setting the control bits as described in **Table 26.40, Control Bit Settings in Independent Channel Output Mode 1**. | Channel operation is stopped. |
| Start operation | Set TAUJnTS.TAUJnTSm to 1.<br>TAUJnTS.TAUJnTSm is a trigger bit, so it is automatically cleared to 0. | TAUJnTE.TAUJnTEm is set to 1 and the counter starts.<br>TAUJnCNTm loads the TAUJnCDRm value.<br>When TAUJnCMORm.TAUJnMD0 = 1, INTTAUJnIm is generated and TAUJTTOUTm toggles. |
| During operation | The TAUJnCDRm register value can be changed at any time.<br>The TAUJnCNTm register can be read at all times. | TAUJnCNTm counts down. When the counter reaches 0000 $0000_H$:<br>• TAUJnCNTm reloads the TAUJnCDRm value and continues count operation.<br>• INTTAUJnIm is generated and TAUJTTOUTm toggles. |
| Stop operation | Set TAUJnTT.TAUJnTTm to 1.<br>TAUJnTT.TAUJnTTm is a trigger bit, so it is automatically cleared to 0. | TAUJnTE.TAUJnTEm is cleared to 0 and the counter stops.<br>TAUJnCNTm and TAUJTTOUTm stop and retain their current values. |

Restart operation

### 26.12.1.6 Specific Timing Diagrams

#### (1) TAUJnCDRm = 0000 $0000_H$, count clock = PCLK/2

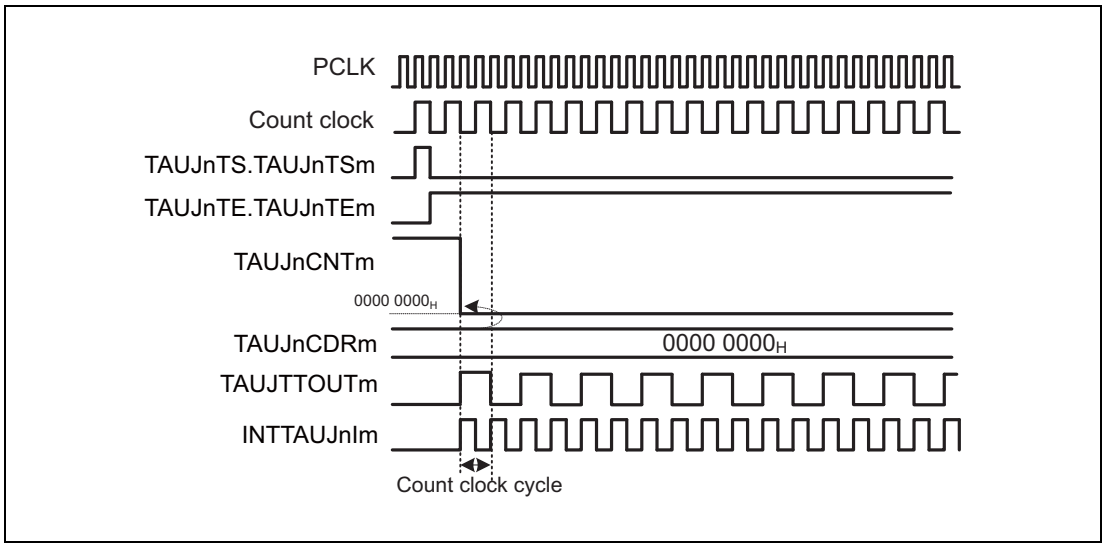



Figure 26.17 TAUJnCDRm = 0000 $0000_H$, Count Clock = PCLK/2

- If TAUJnCDRm = 0000 $0000_H$ and the count clock = PCLK/2, the TAUJnCDRm value is loaded to TAUJnCNTm every count clock, meaning that TAUJnCNTm is always 0000 $0000_H$.
- INTTAUJnIm is generated every count clock, resulting in TAUJTTOUTm toggling every count clock.

### (2) TAUJnCDRm = 0000 0000$_H$, count clock = PCLK

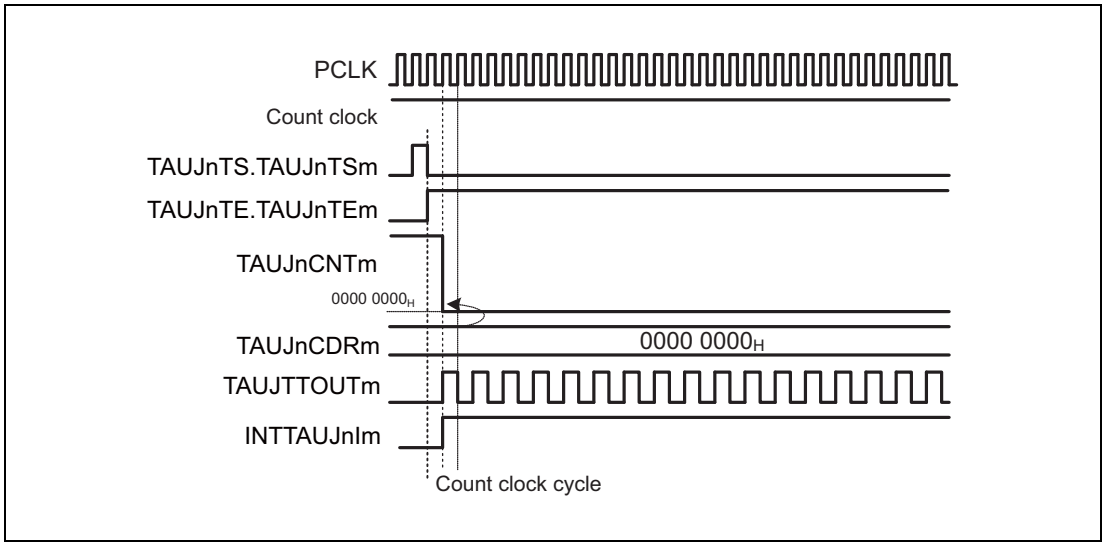



Figure 26.18 TAUJnCDRm = 0000 0000$_H$, Count Clock = PCLK

- If TAUJnCDRm = 0000 0000$_H$ and the count clock = PCLK, the TAUJnCDRm value is loaded to TAUJnCNTm every PCLK clock, meaning that TAUJnCNTm is always 0000 0000$_H$.
- INTTAUJnIm is fixed to the high level. Though the first interrupt is generated, subsequent interrupts are not generated.
  TAUJTTOUTm is toggled every PCLK clock.

### (3) Operation stop and restart

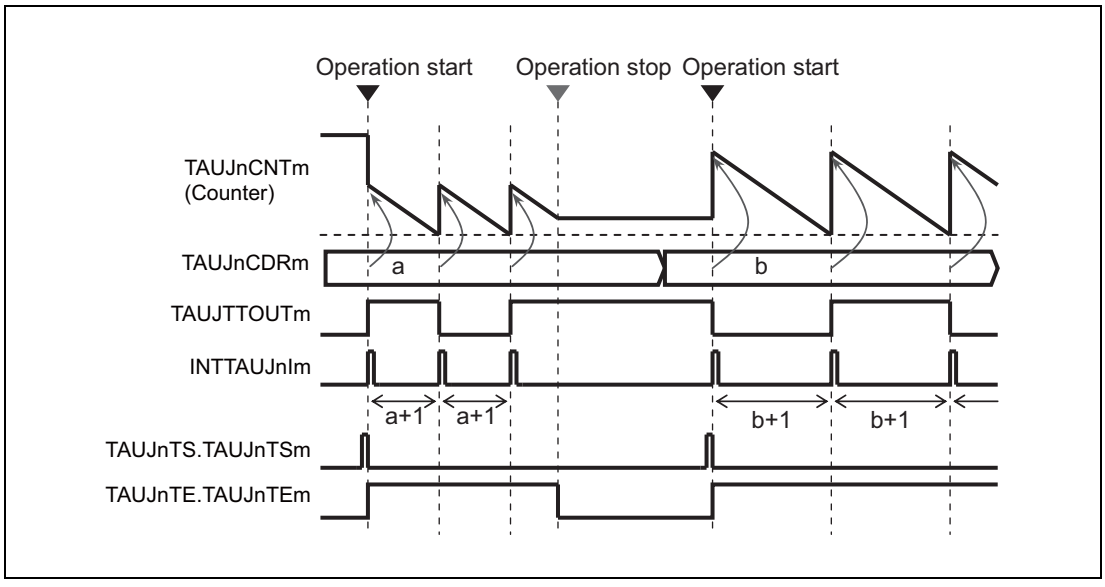



Figure 26.19 Operation Stop and Restart (TAUJnCMORm.TAUJnMD0 = 1)

- The counter can be stopped by setting TAUJnTT.TAUJnTTm to 1, which in turn sets TAUJnTE.TAUJnTEm to 0.
- TAUJnCNTm and TAUJTTOUTm stop but retain their values.
- The counter can be restarted by setting TAUJnTS.TAUJnTSm to 1.

**(4) Forced restart**

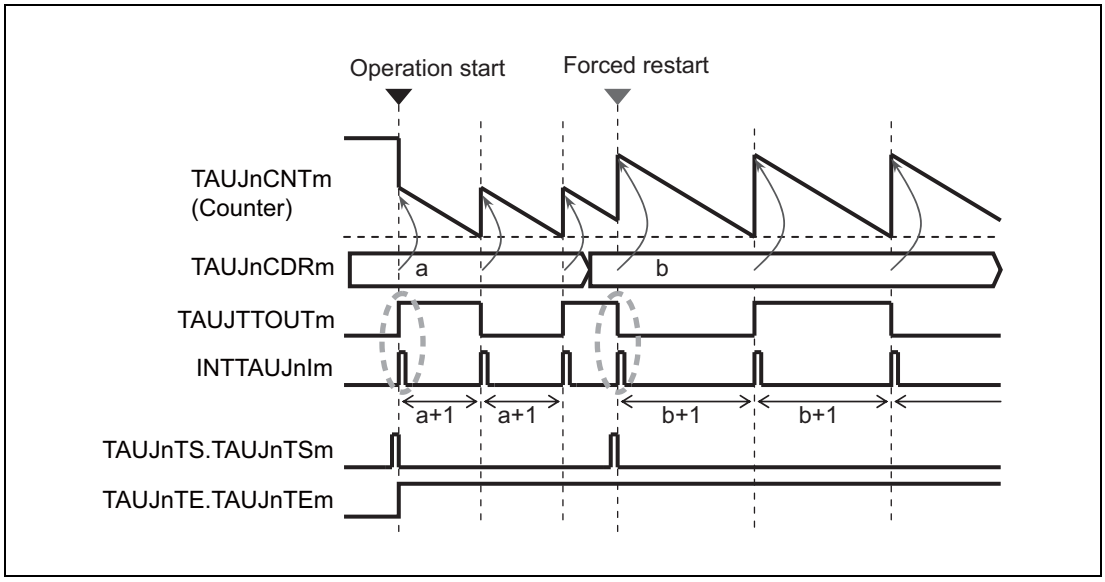



**Figure 26.20 Forced Restart Operation (TAUJnCMORm.TAUJnMD0 = 1)**

- The counter can be forcibly restarted (without stopping it first) by setting TAUJnTS.TAUJnTSm to 1 during operation.
- If the TAUJnCMORm.TAUJnMD0 bit is set to 1, the first interrupt after a start or restart is generated.
- When a forced restart is made, the TAUJnCDRm value is reflected to TAUJnCNTm and counting starts. Execute a forced restart to reflect the changed TAUJnCDRm value immediately.

## 26.12.2 TAUJTTINm Input Interval Timer Function

### 26.12.2.1 Overview

#### Summary

This function is used as a reference timer for generating timer interrupts (INTTAUJnIm) at regular intervals or when a valid TAUJTTINm input edge is detected. When an interrupt is generated, the TAUJTTOUTm signal toggles, resulting in a square wave. Output of square waves is only supported for TAUJ0.

#### Description

This function operates in an identical manner to the interval timer function (see **Section 26.12.1, Interval Timer Function**), except that this function is restarted by a valid TAUJTTINm input edge. The type of edge used as the trigger is specified using the TAUJnCMURm.TAUJnTIS[1:0] bits. Either rising edge, falling edge, or rising and falling edges can be selected.

### 26.12.2.2 Equations

INTTAUJnIm cycle = count clock cycle × (TAUJnCDRm + 1)

TAUJTTOUTm square wave cycle = count clock cycle × (TAUJnCDRm + 1) × 2

### 26.12.2.3 Block Diagram and General Timing Diagram

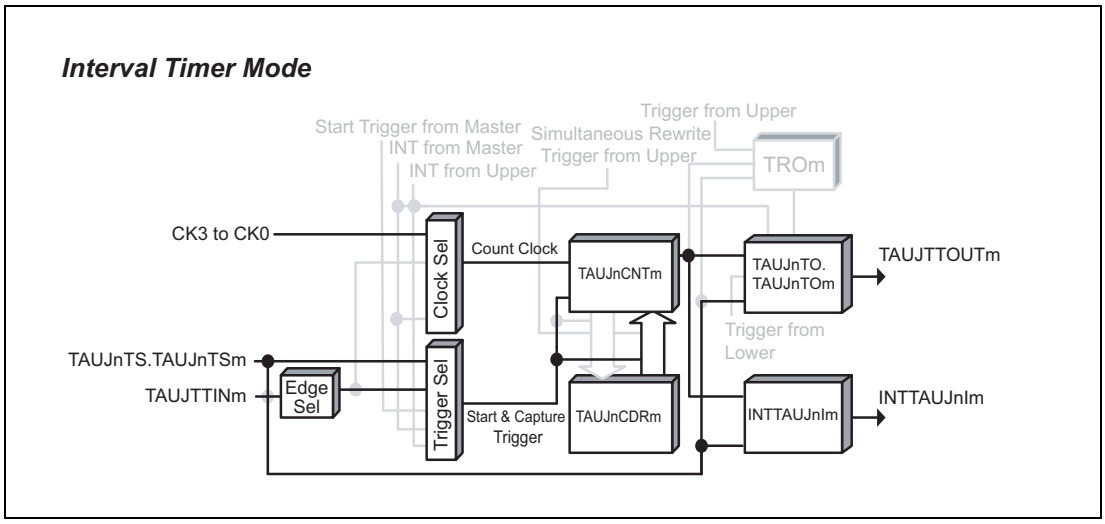



**Figure 26.21 Block Diagram for TAUJTTINm Input Interval Timer Function**

The following settings apply to the general timing diagram.

- INTTAUJnIm is generated at operation start (TAUJnCMORm.TAUJnMD0 = 1).
- Rising edge detection (TAUJnCMURm.TAUJnTIS[1:0] = $01_B$)

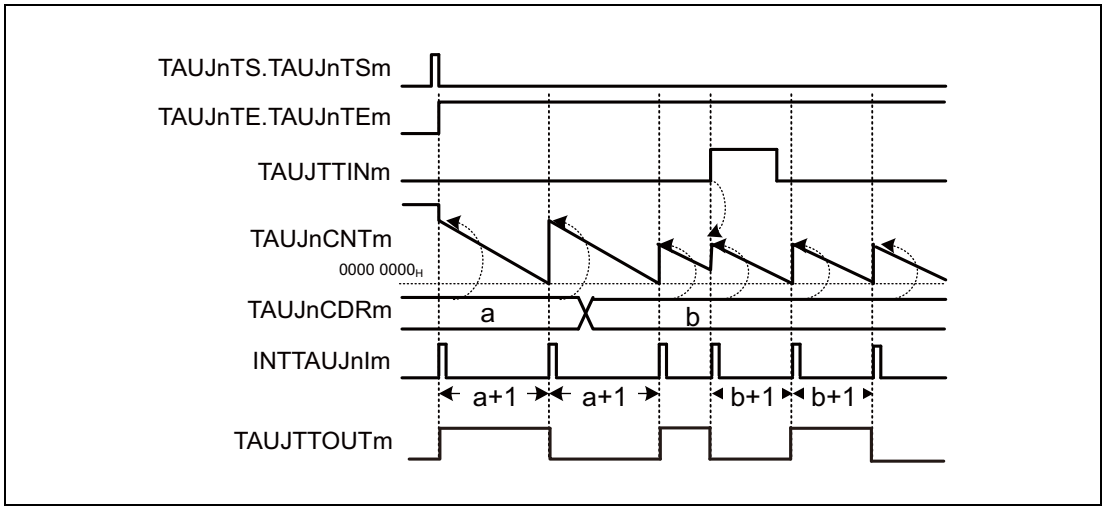



**Figure 26.22 General Timing Diagram for TAUJTTINm Input Interval Timer Function**

### 26.12.2.4 Register Settings

#### (1) TAUJnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUJnCKS[1:0] | | TAUJnCCS[1:0] | | TAUJn MAS | TAUJnSTS[2:0] | | | TAUJnCOS[1:0] | | — | TAUJnMD[4:1] | | | | TAUJn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 26.43 Contents of the TAUJnCMORm register for TAUJTTINm Input Interval Timer Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUJnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output CK0<br>01: Prescaler output CK1<br>10: Prescaler output CK2<br>11: Prescaler output CK3 |
| 13, 12 | TAUJnCCS[1:0] | Write $00_B$. |
| 11 | TAUJnMAS | Write $0_B$. |
| 10 to 8 | TAUJnSTS[2:0] | Write $001_B$. |
| 7, 6 | TAUJnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUJnMD[4:1] | Write $0000_B$. |
| 0 | TAUJnMD0 | 0: INTTAUJnIm is not generated and TAUJTTOUTm does not toggle when operation starts.<br>1: Generates INTTAUJnIm and toggles TAUJTTOUTm when operation starts. |

#### (2) TAUJnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUJnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 26.44 Contents of the TAUJnCMURm register for TAUJTTINm Input Interval Timer Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUJnTIS[1:0] | 00: Falling edge detection<br>01: Rising edge detection<br>10: Rising and falling edge detection<br>11: Setting prohibited |

### (3) Channel output mode

Table 26.45 Control Bit Settings for Independent Channel Output Mode 1

| Bit name | Setting |
|---|---|
| TAUJnTOE.TAUJnTOEm | Write $1_B$. |
| TAUJnTOM.TAUJnTOMm | Write $0_B$. |
| TAUJnTOC.TAUJnTOCm | Write $0_B$. |
| TAUJnTOL.TAUJnTOLm | Write $0_B$. |

**NOTE**

The channel output mode can also be set to channel output mode controlled by software by setting TAUJnTOE.TAUJnTOEm = 0. TAUJTTOUTm can then be controlled independently of the interrupts. For details see **Section 26.7, Channel Output Modes**.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUJnRDE and TAUJnRDM) cannot be used with the TAUJTTINm input interval timer function. Therefore, these registers must be set to 0.

Table 26.46 Simultaneous Rewrite Settings for TAUJTTINm Input Interval Timer Function

| Bit Name | Setting |
|---|---|
| TAUJnRDE.TAUJnRDEm | 0: Disables simultaneous rewrite. |
| TAUJnRDM.TAUJnRDMm | 0: When simultaneous rewrite is disabled (TAUJnRDE.TAUJnRDEm = 0), set these bits to 0. |

### 26.12.2.5 Operating Procedure for TAUJTTINm Input Interval Timer Function

Table 26.47 Operating Procedure for TAUJTTINm Input Interval Timer Function

| | Operation | Status of TAUJn |
|---|---|---|
| Initial channel setting | Set the TAUJnCMORm and TAUJnCMURm registers as described in **Table 26.43, Contents of the TAUJnCMORm register for TAUJTTINm Input Interval Timer Function** and **Table 26.44, Contents of the TAUJnCMURm register for TAUJTTINm Input Interval Timer Function**.<br><br>Set the value of the TAUJnCDRm register<br><br>Set the channel output mode by setting the control bits as described in **Table 26.45, Control Bit Settings for Independent Channel Output Mode 1**. | Channel operation is stopped. |
| Start operation | Set TAUJnTS.TAUJnTSm to 1.<br>TAUJnTS.TAUJnTSm is a trigger bit, so it is automatically cleared to 0. | TAUJnTE.TAUJnTEm is set to 1 and the counter starts.<br>TAUJnCNTm loads the TAUJnCDRm value.<br>When TAUJnCMORm.TAUJnMD0 = 1, INTTAUJnIm is generated and TAUJTTOUTm toggles. |
| During operation | The values of the TAUJnCMURm.TAUJnTIS[1:0] and TAUJnCDRm registers can be changed at any time.<br>The TAUJnCNTm register can be read at all times.<br><br>Detection of TAUJTTINm edge | TAUJnCNTm counts down. When the counter reaches 0000 $0000_H$:<br>• TAUJnCNTm reloads the TAUJnCDRm value and continues count operation.<br>• INTTAUJnIm is generated and TAUJTTOUTm toggles.<br>When a TAUJTTINm input valid edge is detected during count operation, TAUJnCNTm reloads the TAUJnCDRm value and continues count operation.<br>Afterwards, this procedure is repeated. |
| Stop operation | Set TAUJnTT.TAUJnTTm to 1.<br>TAUJnTT.TAUJnTTm is a trigger bit, so it is automatically cleared to 0. | TAUJnTE.TAUJnTEm is cleared to 0 and the counter stops.<br>TAUJnCNTm and TAUJTTOUTm stop and retain their current values. |

Restart operation

### 26.12.2.6 Specific Timing Diagrams

The timing diagrams in **Section 26.12.1, Interval Timer Function** apply, and in addition the counter can also be restarted by a valid TAUJTTINm input edge.

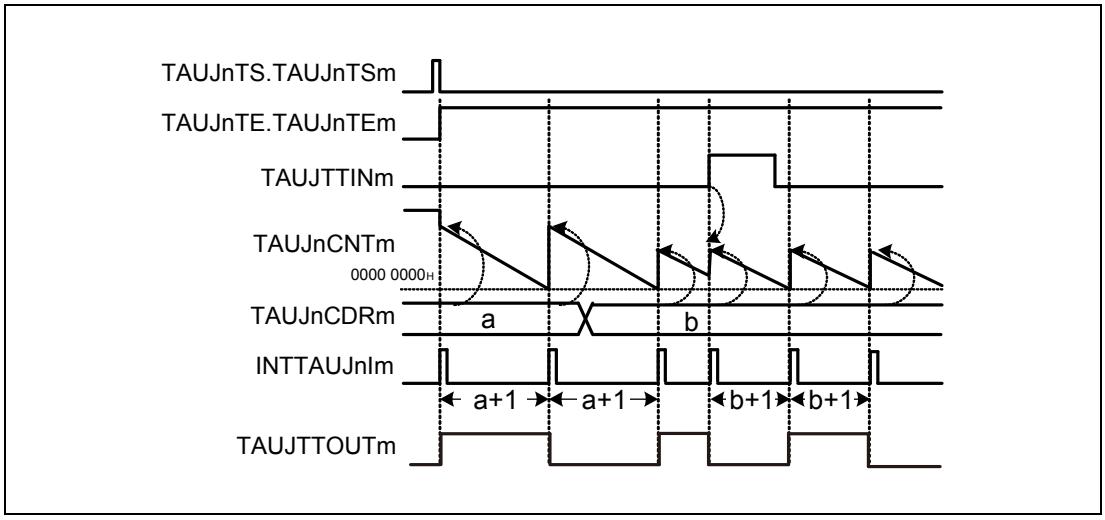



**Figure 26.23 Counter Triggered By Rising TAUJTTINm input edge (TAUJnCMURm.TAUJnTIS[1:0] = $01_B$), TAUJnCMORm.TAUJnMD0 = 1**

If a valid TAUJTTINm input edge is detected, an interrupt is generated which causes TAUJTTOUTm to toggle. In this example, the valid edge is a rising edge (TAUJnCMURm.TAUJnTIS[1:0] = $01_B$).

## 26.12.3 TAUJTTINm Input Pulse Interval Measurement Function

### 26.12.3.1 Overview

#### Summary

This function captures the count value and uses this value and the overflow bit TAUJnCSRm.TAUJnOVF to measure the interval of the TAUJTTINm input signals.

#### Prerequisites

TAUJTTOUTm is not used for this function.

#### Description

The counter is enabled by setting the channel trigger bit (TAUJnTS.TAUJnTSm) to 1. This in turn sets TAUJnTE.TAUJnTEm = 1, enabling count operation. The counter TAUJnCNTm starts counting up from 0000 $0000_H$. When a valid TAUJTTINm edge is detected, the value of TAUJnCNTm is captured, transferred to TAUJnCDRm, and an interrupt INTTAUJnIm is generated. The counter resets to 0000 $0000_H$ and subsequently continues operation.

If the counter reaches FFFF $FFFF_H$ before a valid TAUJTTINm edge is detected, it overflows to 0000 $0000_H$. The counter is reset to 0000 $0000_H$ and subsequently continues operation. The values transferred to TAUJnCDRm and TAUJnCSRm.TAUJnOVF respectively depend on the values of bits TAUJnCMORm.TAUJnCOS[1:0].

**Table 26.48 Effects of an Overflow**

| TAUJnCMORm.TAUJnCOS[1:0] | When Overflow Occurs | | When a Valid TAUJTTINm Input is then Detected | |
|---|---|---|---|---|
| | TAUJnCDRm | TAUJnCSRm.TAUJnOVF | TAUJnCDRm and TAUJnCNTm | TAUJnCSRm.TAUJnOVF |
| 00 | Unchanged | 0 | TAUJnCNTm loaded to TAUJnCDRm | 1 |
| 01 | | 1 | | |
| 10 | Set to FFFF $FFFF_H$ | 0 | TAUJnCNTm set to 0, TAUJnCDRm unchanged | Unchanged |
| 11 | | 1 | | |

When TAUJnCMORm.TAUJnCOS[0] = 1, the overflow bit TAUJnCSRm.TAUJnOVF can only be cleared by setting TAUJnCSCm.TAUJnCLOV = 1.

The combination of the values of TAUJnCDRm and TAUJnCSRm.TAUJnOVF can be used to deduce the interval of the TAUJTTINm signal. However, if an overflow occurs multiple times before a valid TAUJTTINm input is detected, the overflow bit TAUJnCSRm.TAUJnOVF cannot indicate this.

The function can be stopped by setting TAUJnTT.TAUJnTTm = 1, which in turn sets TAUJnTE.TAUJnTEm = 0. TAUJnCNTm stops but retains its value. While the function is stopped, TAUJTTINm input valid edge detection and TAUJnCNTm capture are not performed.

#### Conditions

If the TAUJnCMORm.TAUJnMD0 bit is set to 0, the first interrupt after a start or restart is not generated. For details, see **26.9, TAUJTTOUTm Output and INTTAUJnIm Generation when Counter Starts or Restarts**.

**NOTE**

When TAUJnCMORm.TAUJnCOS[1] = 1, the value of TAUJnCNTm is not loaded to TAUJnCDRm when the first valid TAUJTTINm input edge occurs after an overflow. However, an interrupt is generated.

### 26.12.3.2 Equations

TAUJTTINm input pulse interval = count clock cycle ×
[(TAUJnCSRm.TAUJnOVF × (FFFF $FFFF_H$ + 1)) + TAUJnCDRm capture value + 1]

### 26.12.3.3 Block Diagram and General Timing Diagram

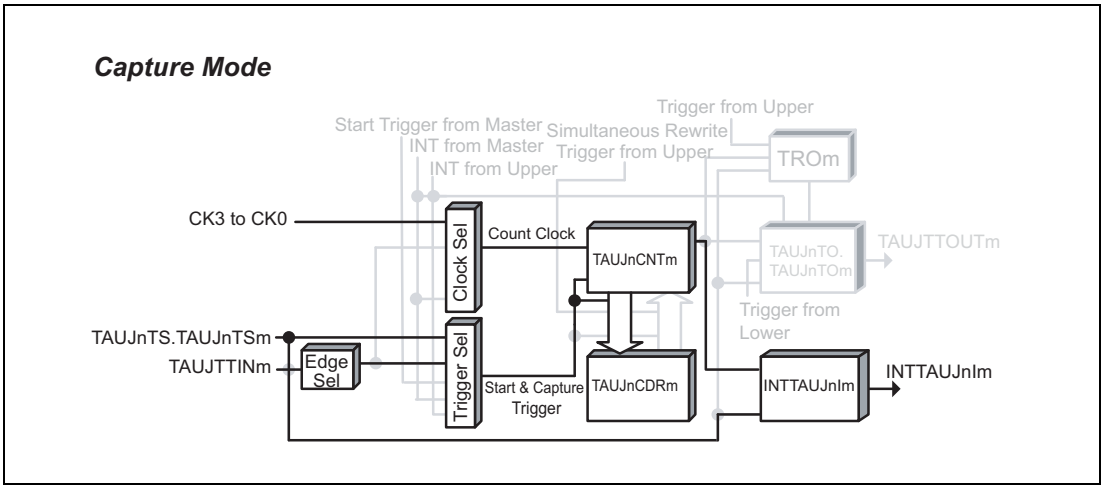



**Figure 26.24 Block Diagram for TAUJTTINm Input Pulse Interval Measurement Function**

The following settings apply to the general timing diagram.

- INTTAUJnIm is not generated when operation starts (TAUJnCMORm.TAUJnMD0 = 0).
- Falling edge detection (TAUJnCMURm.TAUJnTIS[1:0] = $00_B$)
- When a valid TAUJTTINm input is detected after an overflow, TAUJnCDRm is changed and TAUJnCSRm.TAUJnOVF is set to 1 (TAUJnCMORm.TAUJnCOS[1:0] = $00_B$).

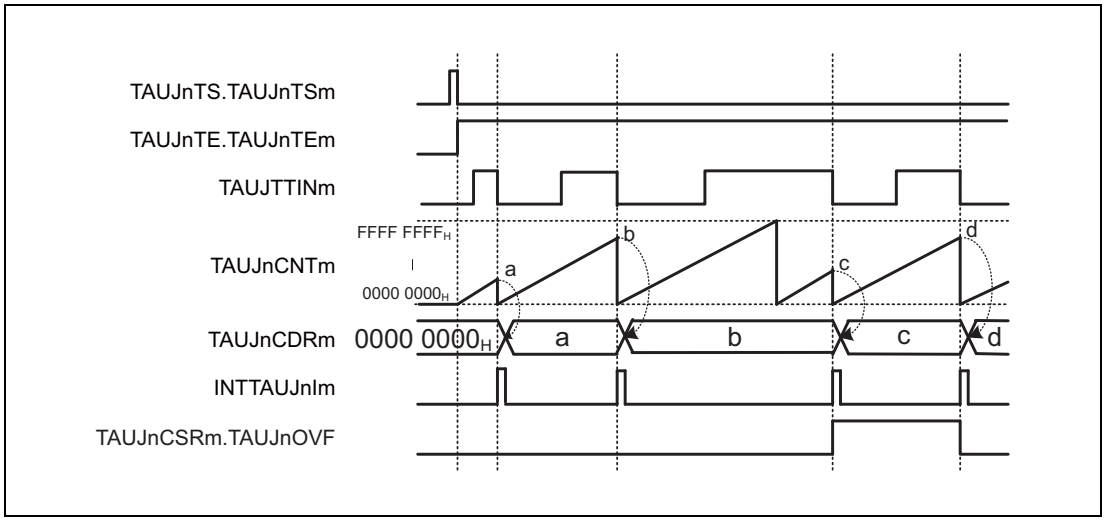



**Figure 26.25 General Timing Diagram For TAUJTTINm Input Pulse Interval Measurement Function**

### 26.12.3.4 Register Settings

#### (1) TAUJnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUJnCKS[1:0] | | TAUJnCCS[1:0] | | TAUJn MAS | TAUJnSTS[2:0] | | | TAUJnCOS[1:0] | | — | TAUJnMD[4:1] | | | | TAUJn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 26.49 Contents of the TAUJnCMORm Register for TAUJTTINm Input Pulse Interval Measurement Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUJnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUJnCCS[1:0] | Write $00_B$. |
| 11 | TAUJnMAS | Write $0_B$. |
| 10 to 8 | TAUJnSTS[2:0] | Write $001_B$. |
| 7, 6 | TAUJnCOS[1:0] | See **Table 26.48, Effects of an Overflow** |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUJnMD[4:1] | Write $0010_B$. |
| 0 | TAUJnMD0 | 0: INTTAUJnIm is not generated when operation starts.<br>1: Generates INTTAUJnIm when operation starts. |

#### (2) TAUJnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUJnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 26.50 Contents of the TAUJnCMURm Register for TAUJTTINm Input Pulse Interval Measurement Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUJnTIS[1:0] | 00: Falling edge detection<br>01: Rising edge detection<br>10: Rising and falling edge detection<br>11: Setting prohibited |

#### (3) Channel output mode

TAUJnTOE.TAUJnTOEm is set to 0 because the channel output mode is not used by this function.

#### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUJnRDE and TAUJnRDM) cannot be used with the TAUJTTINm input pulse interval measurement function. Therefore, these registers must be set to 0.

**Table 26.51 Simultaneous Rewrite Settings for TAUJTTINm Input Pulse Interval Measurement Function**

| Bit name | Setting |
|---|---|
| TAUJnRDE.TAUJnRDEm | 0: Disables simultaneous rewrite. |
| TAUJnRDM.TAUJnRDMm | 0: When simultaneous rewrite is disabled (TAUJnRDE.TAUJnRDEm = 0), set these bits to 0. |

### 26.12.3.5 Operating Procedure for TAUJTTINm Input Pulse Interval Measurement Function

**Table 26.52 Operating Procedure for TAUJTTINm Input Pulse Interval Measurement Function**

| | Operation | Status of TAUJn |
|---|---|---|
| Initial channel setting | Set the TAUJnCMORm and TAUJnCMURm registers as described in **Table 26.49, Contents of the TAUJnCMORm Register for TAUJTTINm Input Pulse Interval Measurement Function** and **Table 26.50, Contents of the TAUJnCMURm Register for TAUJTTINm Input Pulse Interval Measurement Function**.<br><br>The TAUJnCDRm register functions as a capture register. | Channel operation is stopped. |
| Start operation | Set TAUJnTS.TAUJnTSm to 1.<br>TAUJnTS.TAUJnTSm is a trigger bit, so it is automatically cleared to 0. | TAUJnTE.TAUJnTEm is set to 1 and the counter starts.<br>TAUJnCNTm is cleared to 0000 $0000_H$.<br>INTTAUJnIm is generated when TAUJnCMORm.TAUJnMD0 is set to 1. |
| During operation | Detection of TAUJTTINm edges.<br><br>The values of the TAUJnCMURm.TAUJnTIS[1:0] bits can be changed at any time.<br>The TAUJnCDRm and TAUJnCSRm registers can be read at any time.<br>TAUJnCSCm.TAUJnCLOV bit can be written to 1. (TAUJnCSRm.TAUJnOVF bit is cleared to 0.) | TAUJnCNTm starts to count up from 0000 $0000_H$.<br>When a TAUJTTINm valid edge is detected:<br>• TAUJnCNTm transfers (captures) its value to TAUJnCDRm, and returns to 0000 $0000_H$.<br>• INTTAUJnIm is then generated.<br>Afterwards, this procedure is repeated. |
| Stop operation | Set TAUJnTT.TAUJnTTm to 1.<br>TAUJnTT.TAUJnTTm is a trigger bit, so it is automatically cleared to 0. | TAUJnTE.TAUJnTEm is cleared to 0 and the counter stops.<br>TAUJnCNTm stops and both it and TAUJnCSRm.TAUJnOVF retain their current values. |

Restart operation

### 26.12.3.6 Specific Timing Diagrams: Overflow Behavior

#### (1) TAUJnCMORm.TAUJnCOS[1:0] = $00_B$

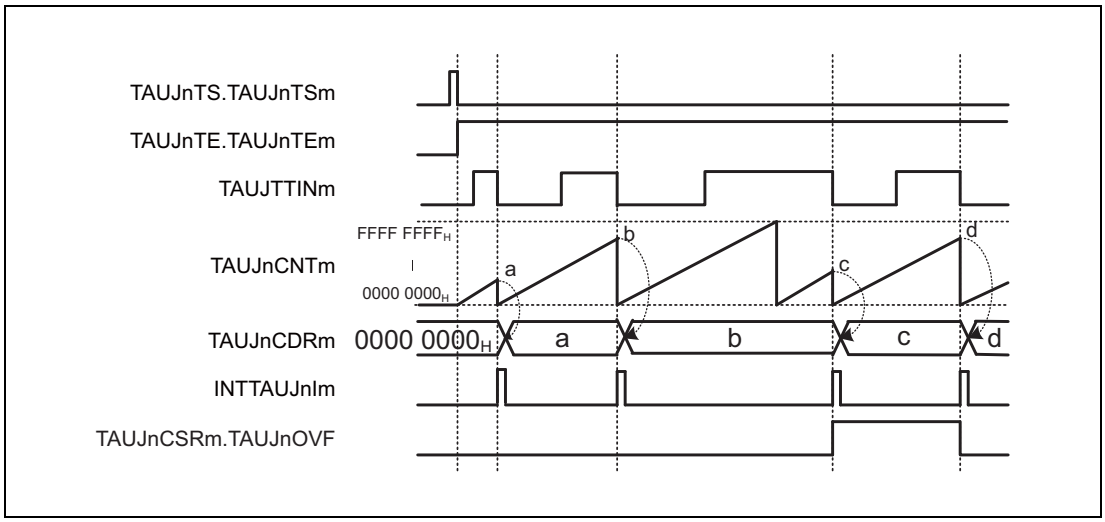



**Figure 26.26 TAUJnCMORm.TAUJnCOS[1:0] = $00_B$, TAUJnCMORm.TAUJnMD0 = 0, TAUJnCMURm.TAUJnTIS[1:0] = $00_B$**

- When an overflow occurs, the value of TAUJnCDRm remains unchanged and the value of TAUJnCSRm.TAUJnOVF remains 0.
- Upon detection of the next valid TAUJTTINm input edge, the value of TAUJnCNTm is loaded to TAUJnCDRm and TAUJnCSRm.TAUJnOVF is set to 1.
- If the next valid TAUJTTINm input edge is detected when no overflow occurs, TAUJnCSRm.TAUJnOVF is cleared to 0.

### (2) TAUJnCMORm.TAUJnCOS[1:0] = $01_B$

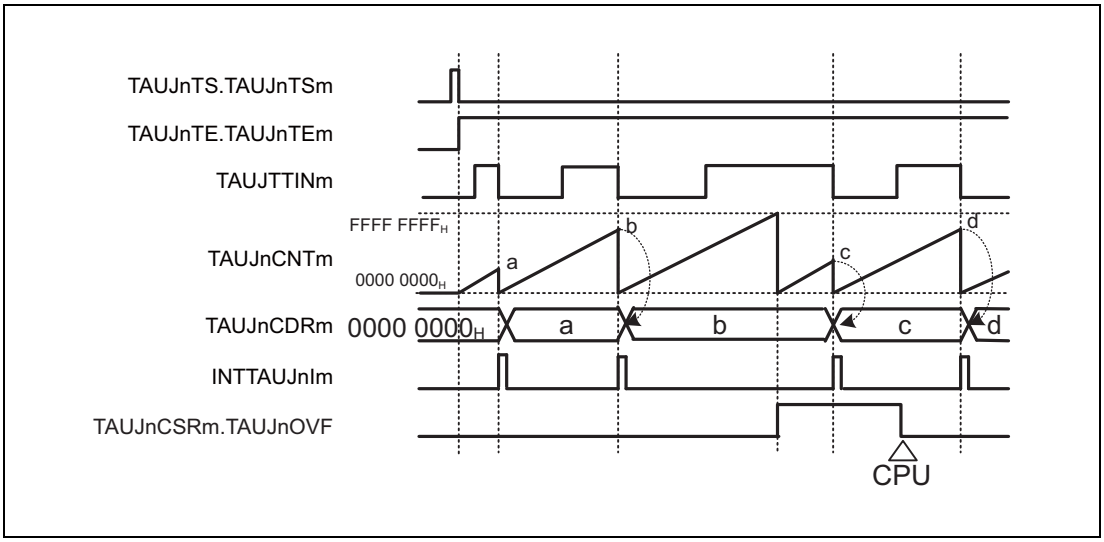



**Figure 26.27 TAUJnCMORm.TAUJnCOS[1:0] = $01_B$, TAUJnCMORm.TAUJnMD0 = 0, TAUJnCMURm.TAUJnTIS[1:0] = $00_B$**

- When an overflow occurs, the value of TAUJnCDRm remains unchanged and TAUJnCSRm.TAUJnOVF is set to 1.
- Upon detection of the next valid TAUJTTINm input edge, the value of TAUJnCNTm is loaded to TAUJnCDRm.
- TAUJnCSRm.TAUJnOVF is only cleared by a CPU command (by setting the TAUJnCSCm.TAUJnCLOV bit to 1).

### (3) TAUJnCMORm.TAUJnCOS[1:0] = $10_B$

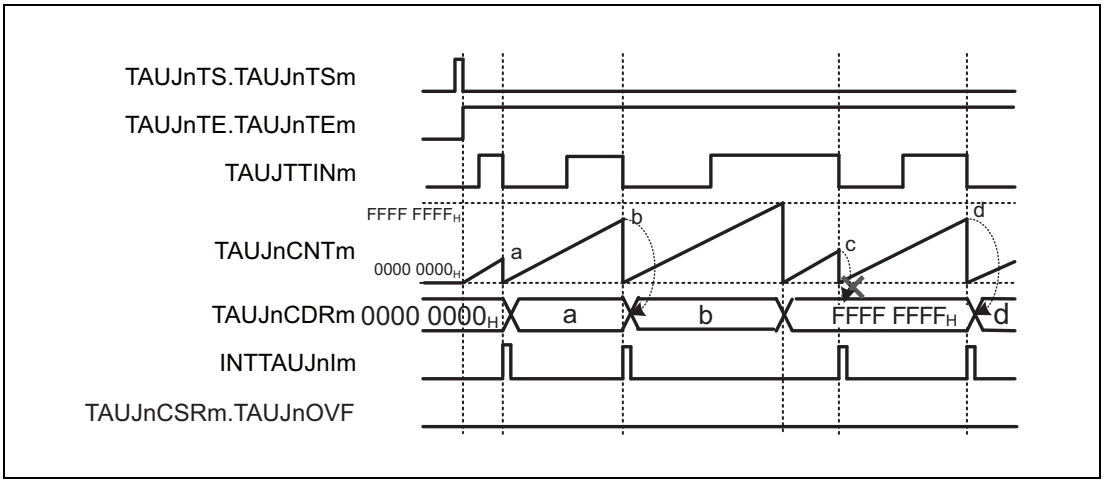



**Figure 26.28 TAUJnCMORm.TAUJnCOS[1:0] = $10_B$, TAUJnCMORm.TAUJnMD0 = 0, TAUJnCMURm.TAUJnTIS[1:0] = $00_B$**

- When an overflow occurs, TAUJnCDRm is set to FFFF FFFF$_H$ and the value of TAUJnCSRm.TAUJnOVF remains 0.
- Upon detection of the next valid TAUJTTINm input edge, TAUJnCNTm is reset to 0, but TAUJnCDRm and TAUJnCSRm.TAUJnOVF remain unchanged.
- Thus, the next TAUJTTINm input valid edge after the overflow is ignored.

### (4) TAUJnCMORm.TAUJnCOS[1:0] = $11_B$

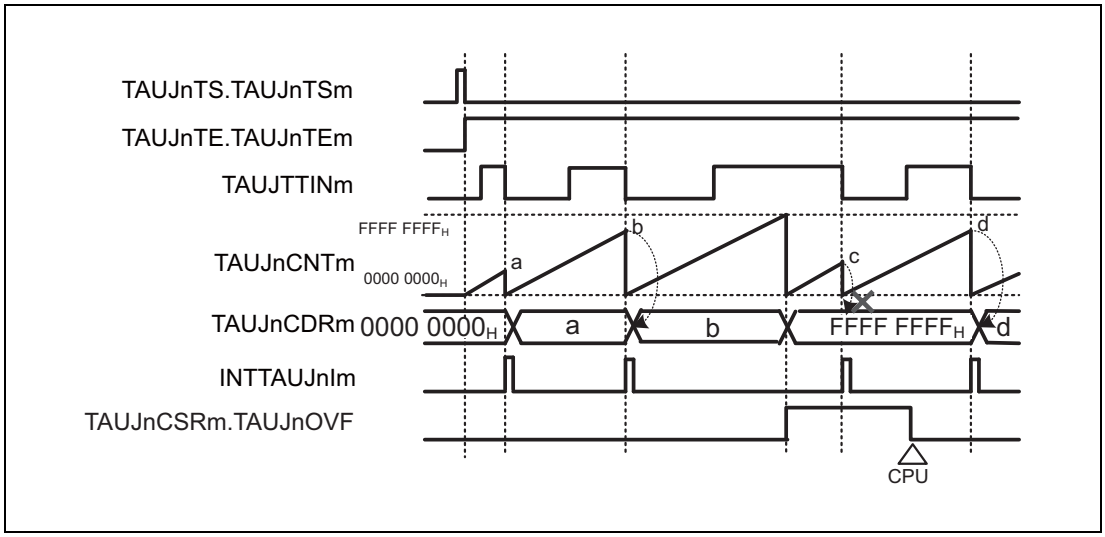



**Figure 26.29 TAUJnCMORm.TAUJnCOS[1:0] = $11_B$, TAUJnCMORm.TAUJnMD0 = 0, TAUJnCMURm.TAUJnTIS[1:0] = $00_B$**

- When an overflow occurs, TAUJnCDRm is set to FFFF FFFF$_H$, and TAUJnCSRm.TAUJnOVF is set to 1.
- Upon detection of the next valid TAUJTTINm input edge, TAUJnCNTm is reset to 0, but TAUJnCDRm and TAUJnCSRm.TAUJnOVF remain unchanged.
- Thus, the next TAUJTTINm input valid edge after the overflow is ignored.
- TAUJnCSRm.TAUJnOVF is cleared by setting the TAUJnCSCm.TAUJnCLOV bit to 1.

## 26.12.4 TAUJTTINm Input Signal Width Measurement Function

### 26.12.4.1 Overview

#### Summary

This function measures the width of a TAUJTTINm signal by starting counting on one edge of the TAUJTTINm signal and capturing the counter value on the opposite edge.

#### Prerequisites

TAUJTTOUTm is not used for this function.

#### Description

The counter is enabled by setting the channel trigger bit (TAUJnTS.TAUJnTSm) to 1. This in turn sets TAUJnTE.TAUJnTEm = 1, enabling count operation. When a valid TAUJTTINm start edge is detected, the counter TAUJnCNTm starts counting up from 0000 $0000_H$. When a valid TAUJTTINm stop edge is detected, the value of TAUJnCNTm is captured, transferred to TAUJnCDRm, and an interrupt INTTAUJnIm is generated. The counter retains its value and awaits the next valid TAUJTTINm input start edge.

If the counter reaches FFFF $FFFF_H$ before a valid TAUJTTINm stop edge is detected, it overflows. The counter is reset to 0000 $0000_H$ and subsequently continues operation. The values transferred to TAUJnCDRm and TAUJnCSRm.TAUJnOVF respectively depend on the values of bits TAUJnCMORm.TAUJnCOS[1:0].

**Table 26.53 Effects of an Overflow**

| TAUJnCMORm. TAUJnCOS[1:0] | When Overflow Occurs | | When a Valid TAUJTTINm Input Stop Edge is Detected | |
|---|---|---|---|---|
| | TAUJnCDRm | TAUJnCSRm. TAUJnOVF | TAUJnCDRm and TAUJnCNTm | TAUJnCSRm. TAUJnOVF |
| 00 | Unchanged | 0 | TAUJnCNTm is loaded to TAUJnCDRm. | 1 |
| 01 | | 1 | | |
| 10 | Set to FFFF $FFFF_H$ | 0 | TAUJnCNTm stops counting, TAUJnCDRm unchanged | Unchanged |
| 11 | | 1 | | |

When TAUJnCMORm.TAUJnCOS[0] = 1, the overflow bit TAUJnCSRm.TAUJnOVF can only be cleared by setting TAUJnCSCm.TAUJnCLOV to 1.

The combination of the values of TAUJnCDRm and TAUJnCSRm.TAUJnOVF can be used to deduce the width of the TAUJTTINm signal. However, if an overflow occurs multiple times before a valid TAUJTTINm input is detected, the overflow bit TAUJnCSRm.TAUJnOVF cannot indicate this.

This function cannot be forcibly restarted.

**NOTE**

When TAUJnCMORm.COS[1] = 1, the value of TAUJnCNTm is not loaded to TAUJnCDRm when the first valid TAUJTTINm input edge occurs after an overflow. However, an interrupt is generated.

### 26.12.4.2 Equations

TAUJTTINm input signal width = count clock cycle ×
[(TAUJnCSRm.TAUJnOVF × (FFFF FFFF$_H$ + 1)) + TAUJnCDRm capture value + 1]

### 26.12.4.3 Block Diagram and General Timing Diagram

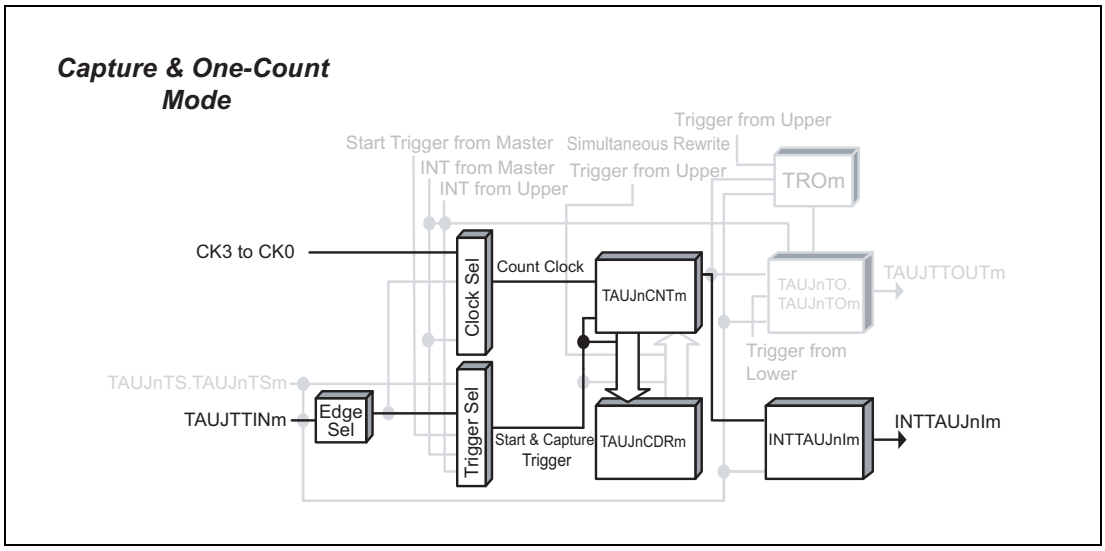



**Figure 26.30 Block Diagram for TAUJTTINm Input Signal Width Measurement Function**

The following settings apply to the general timing diagram.

- Rising and falling edge detection = high width measurement (TAUJnCMURm.TAUJnTIS[1:0] = $11_B$)
- When a valid TAUJTTINm input is detected after an overflow, TAUJnCDRm is changed and TAUJnCSRm.TAUJnOVF is set to 1 (TAUJnCMORm.TAUJnCOS[1:0] = $00_B$).

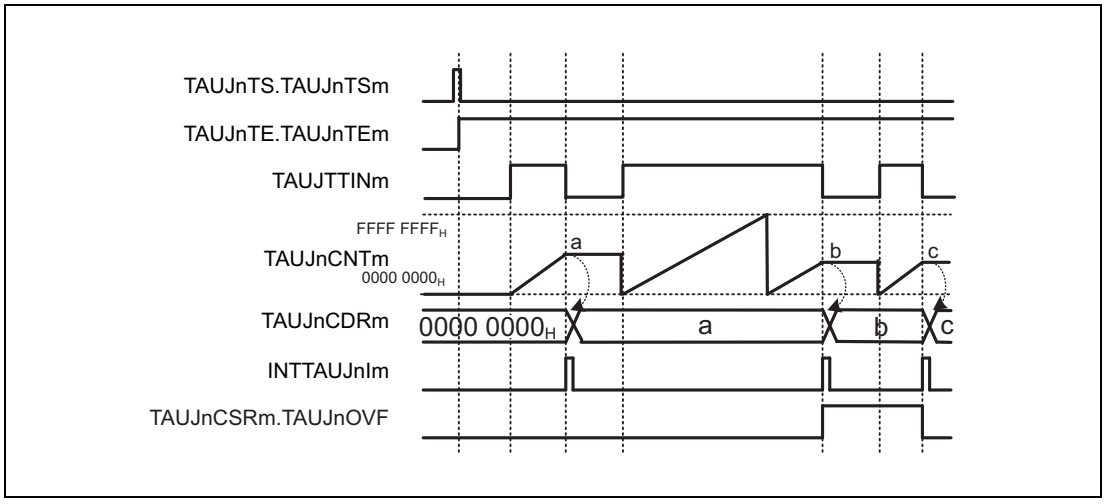



**Figure 26.31 General Timing Diagram for TAUJTTINm Input Signal Width Measurement Function**

### 26.12.4.4 Register Settings

#### (1) TAUJnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUJnCKS[1:0] | | TAUJnCCS[1:0] | | TAUJn MAS | TAUJnSTS[2:0] | | | TAUJnCOS[1:0] | | — | TAUJnMD[4:1] | | | | TAUJn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 26.54 Contents of the TAUJnCMORm Register for TAUJTTINm Input Signal Width Measurement Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUJnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output CK0<br>01: Prescaler output CK1<br>10: Prescaler output CK2<br>11: Prescaler output CK3 |
| 13, 12 | TAUJnCCS[1:0] | Write $00_B$. |
| 11 | TAUJnMAS | Write $0_B$. |
| 10 to 8 | TAUJnSTS[2:0] | Write $010_B$. |
| 7, 6 | TAUJnCOS[1:0] | See **Table 26.53, Effects of an Overflow**. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUJnMD[4:1] | Write $0110_B$. |
| 0 | TAUJnMD0 | Write $0_B$. |

#### (2) TAUJnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUJnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 26.55 Contents of the TAUJnCMURm Register for TAUJTTINm Input Signal Width Measurement Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUJnTIS[1:0] | 10: Rising and falling edge detection (low width measurement)<br>11: Rising and falling edge detection (high width measurement) |

#### (3) Channel output mode

TAUJnTOE.TAUJnTOEm is set to 0 because the channel output mode is not used with this function.

#### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUJnRDE and TAUJnRDM) cannot be used with the TAUJTTINm input signal width measurement function. Therefore, these registers must be set to 0.

**Table 26.56 Simultaneous Rewrite Settings for TAUJTTINm Input Signal Width Measurement Function**

| Bit Name | Setting |
|---|---|
| TAUJnRDE.TAUJnRDEm | 0: Disables simultaneous rewrite. |
| TAUJnRDM.TAUJnRDMm | 0: When simultaneous rewrite is disabled (TAUJnRDE.TAUJnRDEm = 0), set these bits to 0. |

### 26.12.4.5 Operating Procedure for TAUJTTINm Input Signal Width Measurement Function

**Table 26.57 Operating Procedure for TAUJTTINm Input Signal Width Measurement Function**

| | Operation | Status of TAUJn |
|---|---|---|
| Initial channel setting | Set the TAUJnCMORm and TAUJnCMURm registers as described in **Table 26.54, Contents of the TAUJnCMORm Register for TAUJTTINm Input Signal Width Measurement Function** and **Table 26.55, Contents of the TAUJnCMURm Register for TAUJTTINm Input Signal Width Measurement Function**.<br><br>The TAUJnCDRm register functions as a capture register. | Channel operation is stopped. |
| Start operation | Set TAUJnTS.TAUJnTSm to 1.<br>TAUJnTS.TAUJnTSm is a trigger bit, so it is automatically cleared to 0. | TAUJnTE.TAUJnTEm is set to 1 and TAUJnCNTm waits for detection of the TAUJTTINm start edge.<br>When a TAUJTTINm start edge is detected, TAUJnCNTm starts to count up. |
| During operation | The TAUJnCDRm, TAUJnCNTm, and TAUJnCSRm registers can be read at any time.<br>The TAUJnCSCm.TAUJnCLOV bit can be set to 1. | TAUJnCNTm starts to count up from 0000 $0000_H$.<br>When a TAUJTTINm valid edge is detected:<br>• TAUJnCNTm transfers (captures) its value to TAUJnCDRm, and retains its value.<br>• INTTAUJnIm is then generated.<br>• Counting stops at the "value that transferred to TAUJnCDRm + 1" and TAUJnCNTm waits for detection of the TAUJTTINm start edge.<br>Afterwards, this procedure is repeated. |
| Stop operation | Set TAUJnTT.TAUJnTTm to 1.<br>TAUJnTT.TAUJnTTm is a trigger bit, so it is automatically cleared to 0. | TAUJnTE.TAUJnTEm is cleared to 0 and the counter stops.<br>TAUJnCNTm stops and both it and TAUJnCSRm.TAUJnOVF retain their current values. |

Restart operation

### 26.12.4.6 Specific Timing Diagrams: Overflow Behavior

#### (1) TAUJnCMORm.TAUJnCOS[1:0] = $00_B$

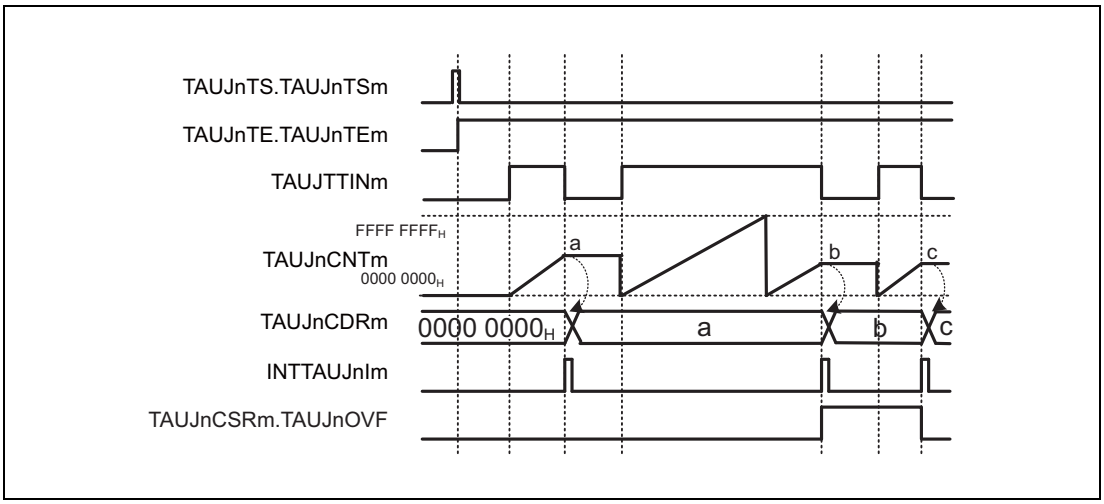



**Figure 26.32 TAUJnCMORm.TAUJnCOS[1:0] = $00_B$, TAUJnCMORm.TAUJnMD0 = 0, TAUJnCMURm.TAUJnTIS[1:0] = $11_B$**

- When an overflow occurs, the value of TAUJnCDRm remains unchanged and the value of TAUJnCSRm.TAUJnOVF remains 0.
- Upon detection of the next valid TAUJTTINm input edge, the value of TAUJnCNTm is loaded to TAUJnCDRm and TAUJnCSRm.TAUJnOVF is set to 1.
- Upon detection of the next valid TAUJTTINm input edge with no overflow occurring, TAUJnCSRm.TAUJnOVF is cleared to 0.

### (2) TAUJnCMORm.TAUJnCOS[1:0] = $01_B$

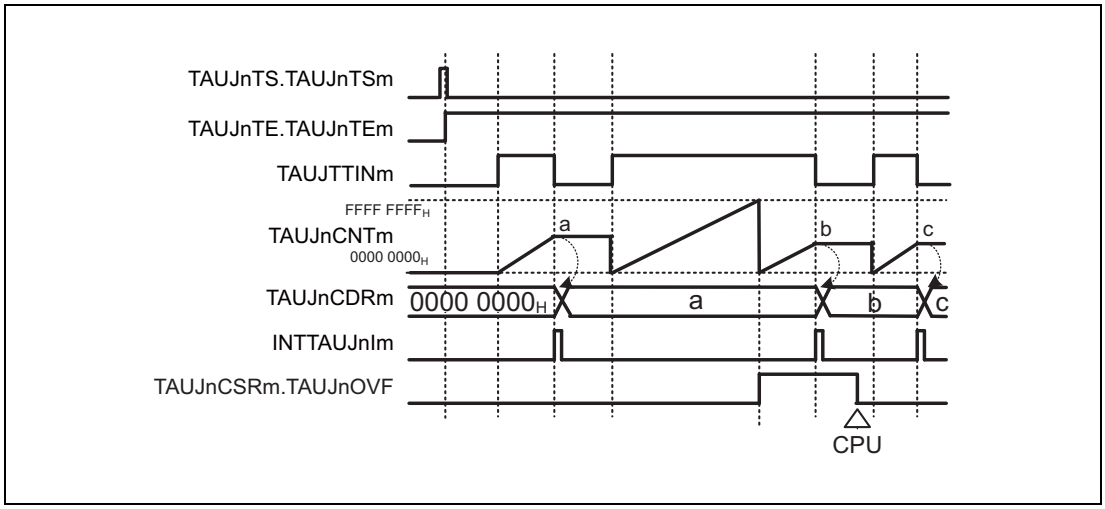



**Figure 26.33 TAUJnCMORm.TAUJnCOS[1:0] = $01_B$, TAUJnCMORm.TAUJnMD0 = 0, TAUJnCMURm.TAUJnTIS[1:0] = $11_B$**

- When an overflow occurs, the value of TAUJnCDRm remains unchanged and the value of TAUJnCSRm.TAUJnOVF is set to 1.
- Upon detection of the next valid TAUJTTINm input edge, the value of TAUJnCNTm is loaded to TAUJnCDRm.
- TAUJnCSRm.TAUJnOVF is only cleared by a CPU command (by setting the TAUJnCSCm.TAUJnCLOV bit to 1).

### (3) TAUJnCMORm.TAUJnCOS[1:0] = $10_B$

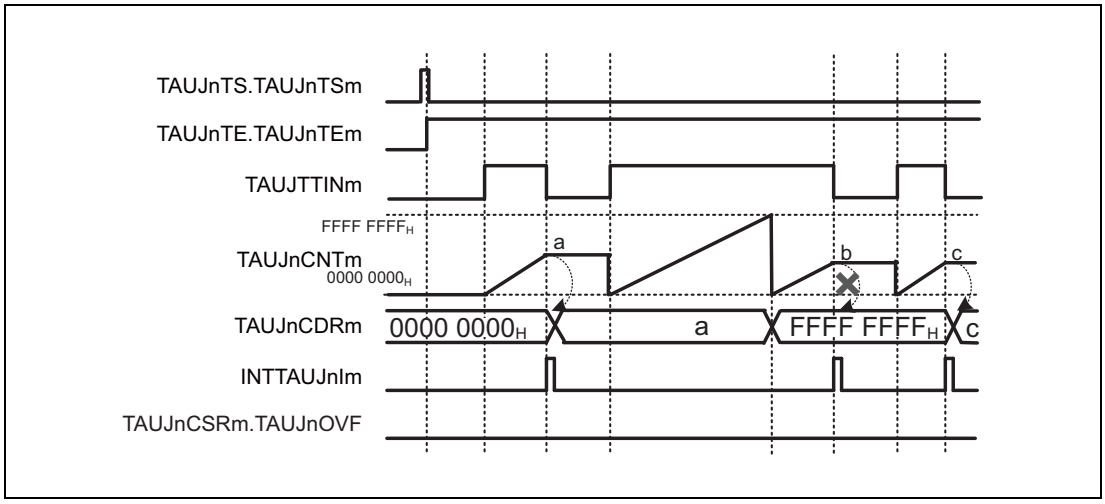



**Figure 26.34 TAUJnCMORm.TAUJnCOS[1:0] = $10_B$, TAUJnCMORm.TAUJnMD0 = 0, TAUJnCMURm.TAUJnTIS[1:0] = $11_B$**

- When an overflow occurs, TAUJnCDRm is set to FFFF FFFF$_H$ and the value of TAUJnCSRm.TAUJnOVF remains 0.
- Upon detection of the next valid TAUJTTINm input edge, TAUJnCNTm stops counting, but TAUJnCDRm and TAUJnCSRm.TAUJnOVF remain unchanged.
- Thus, the next TAUJTTINm input valid edge after the overflow is ignored.

### (4) TAUJnCMORm.TAUJnCOS[1:0] = $11_B$

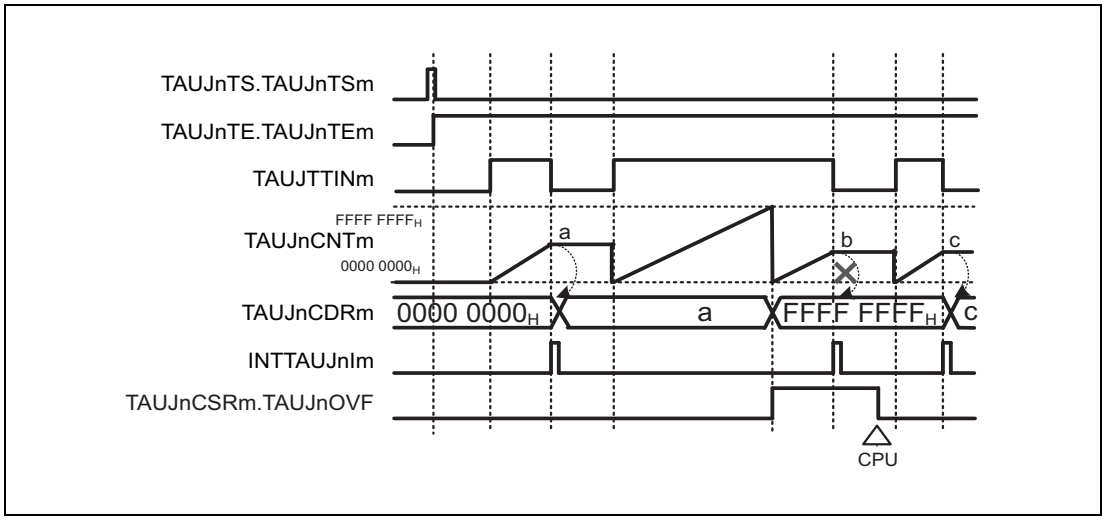



**Figure 26.35 TAUJnCMORm.TAUJnCOS[1:0] = $11_B$, TAUJnCMORm.TAUJnMD0 = 0, TAUJnCMURm.TAUJnTIS[1:0] = $11_B$**

- When an overflow occurs, TAUJnCDRm is set to FFFF FFFF$_H$, and TAUJnCSRm.TAUJnOVF is set to 1.
- Upon detection of the next valid TAUJTTINm input edge, TAUJnCNTm stops counting, but TAUJnCDRm and TAUJnCSRm.TAUJnOVF remain unchanged.
- Thus, the next TAUJTTINm input valid edge after the overflow is ignored.
- TAUJnCSRm.TAUJnOVF is cleared by setting the TAUJnCSCm.TAUJnCLOV bit to 1.

## 26.12.5 TAUJTTINm Input Position Detection Function

### 26.12.5.1 Overview

#### Summary

This function measures the interval of input signals by capturing the counter value on a valid edge of the TAUJTTINm signal.

#### Prerequisites

TAUJTTOUTm is not used for this function

#### Description

The counter is enabled by setting the channel trigger bit (TAUJnTS.TAUJnTSm) to 1. This in turn sets TAUJnTE.TAUJnTEm = 1, enabling count operation. The counter starts to count from 0000 $0000_H$. When a valid TAUJTTINm input edge is detected, the current TAUJnCNTm value is loaded to TAUJnCDRm and an interrupt (INTTAUJnIm) is generated. The counter continues to count.

When the counter reaches FFFF $FFFF_H$, the counter restarts from 0000 $0000_H$.

**NOTE**

The input TAUJTTINm is sampled at the frequency of the operation clock, specified by TAUJnCMORm.TAUJnCKS[1:0] bits. As a result, the output cycle of TAUJTTOUTm has an error of ± 1 operation clock cycle.

#### Conditions

If the TAUJnCMORm. TAUJnMD0 bit is set to 0, the first interrupt after a start or restart is not generated.

### 26.12.5.2 Equations

Function duration at a TAUJTTINm input pulse =
count clock cycle × (TAUJnCDRm capture value + 1)

### 26.12.5.3 Block Diagram and General Timing Diagram

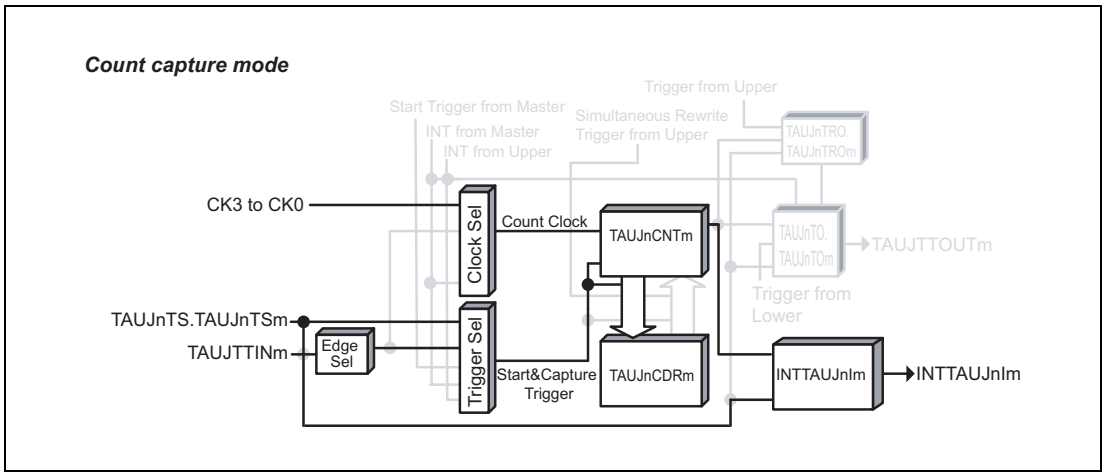



**Figure 26.36 Block Diagram of TAUJTTINm Input Period Count Detection Function**

The following settings apply to the general timing diagram.

- INTTAUJnIm is not generated when operation starts (TAUJnCMORm.TAUJnMD0 = 0).
- Falling edge detection (TAUJnCMURm.TAUJnTIS[1:0] = $00_B$)

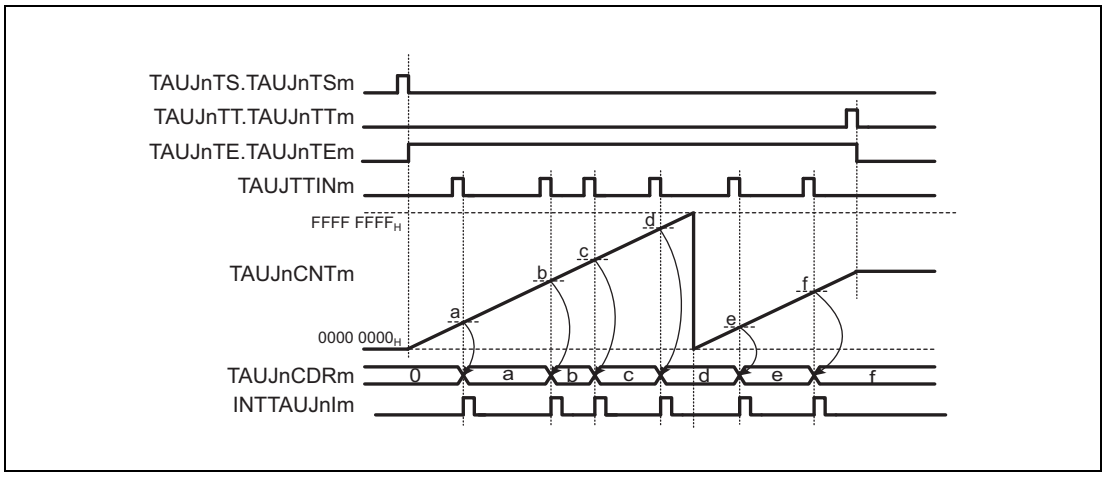



**Figure 26.37 General Timing Diagram for TAUJTTINm Input Position Detection Function**

### 26.12.5.4 Register Settings

#### (1) TAUJnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUJnCKS[1:0] | | TAUJnCCS[1:0] | | TAUJn MAS | TAUJnSTS[2:0] | | | TAUJnCOS[1:0] | | — | TAUJnMD[4:1] | | | | TAUJn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 26.58 Contents of the TAUJnCMORm Register for TAUJTTINm Input Position Detection Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUJnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUJnCCS[1:0] | Write $00_B$. |
| 11 | TAUJnMAS | Write $0_B$. |
| 10 to 8 | TAUJnSTS[2:0] | Write $001_B$. |
| 7, 6 | TAUJnCOS[1:0] | Write $01_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUJnMD[4:1] | Write $1011_B$. |
| 0 | TAUJnMD0 | 0: INTTAUJnIm is not generated when operation starts.<br>1: Generates INTTAUJnIm when operation starts. |

#### (2) TAUJnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUJnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 26.59 Contents of the TAUJnCMURm Register for TAUJTTINm Input Position Detection Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUJnTIS[1:0] | 00: Falling edge detection<br>01: Rising edge detection<br>10: Rising and falling edge detection<br>11: Setting prohibited |

#### (3) Channel output mode

The channel output mode is not used by this function.

#### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUJnRDE and TAUJnRDM) cannot be used with the TAUJTTINm input position detection function. Therefore, these registers must be set to 0.

**Table 26.60 Simultaneous Rewrite Settings for TAUJTTINm Input Position Detection Function**

| Bit Name | Setting |
|---|---|
| TAUJnRDE.TAUJnRDEm | 0: Disables simultaneous rewrite |
| TAUJnRDM.TAUJnRDMm | 0: When simultaneous rewrite is disabled (TAUJnRDE.TAUJnRDEm=0), set these bits to 0 |

### 26.12.5.5 Operating Procedure for TAUJTTINm Input Position Detection Function

**Table 26.61 Operating Procedure for TAUJTTINm Input Position Detection Function**

| | Operation | Status of TAUJn |
|---|---|---|
| Initial channel setting | Set the TAUJnCMORm register and TAUJnCMURm registers as described in **Table 26.58, Contents of the TAUJnCMORm Register for TAUJTTINm Input Position Detection Function** and **Table 26.59, Contents of the TAUJnCMURm Register for TAUJTTINm Input Position Detection Function**.<br><br>The TAUJnCDRm register functions as a capture register. | Channel operation is stopped. |
| Start operation | Set TAUJnTS.TAUJnTSm to 1.<br>TAUJnTS.TAUJnTSm is a trigger bit, so it is automatically cleared to 0. | TAUJnTE.TAUJnTEm is set to 1 and the counter starts.<br>INTTAUJnIm is generated when TAUJnCMORm.TAUJnMD0 is set to 1. |
| During operation | The values of the TAUJnCMURm.TAUJnTIS[1:0] bits can be changed at any time.<br>The TAUJnCDRm and TAUJnCSRm registers can be read at any time. | TAUJnCNTm starts to count up from 0000 $0000_H$.<br>When a TAUJTTINm valid edge is detected:<br>• TAUJnCNTm transfers (captures) its value to TAUJnCDRm.<br>• INTTAUJnIm is output.<br>• The counter value is not cleared to 0000 $0000_H$ and TAUJnCNTm continues count operation.<br>Afterwards, this procedure is repeated.<br>When TAUJnCNTm reaches FFFF $FFFF_H$, the counter restarts from 0000 $0000_H$. |
| Stop operation | Set TAUJnTT.TAUJnTTm to 1.<br>TAUJnTT.TAUJnTTm is a trigger bit, so it is automatically cleared to 0. | TAUJnTE.TAUJnTEm is cleared to 0 and the counter stops.<br>TAUJnCNTm stops and retains its current value. |

Restart operation

### 26.12.5.6 Specific Timing Diagrams

#### (1) Operation stop and restart

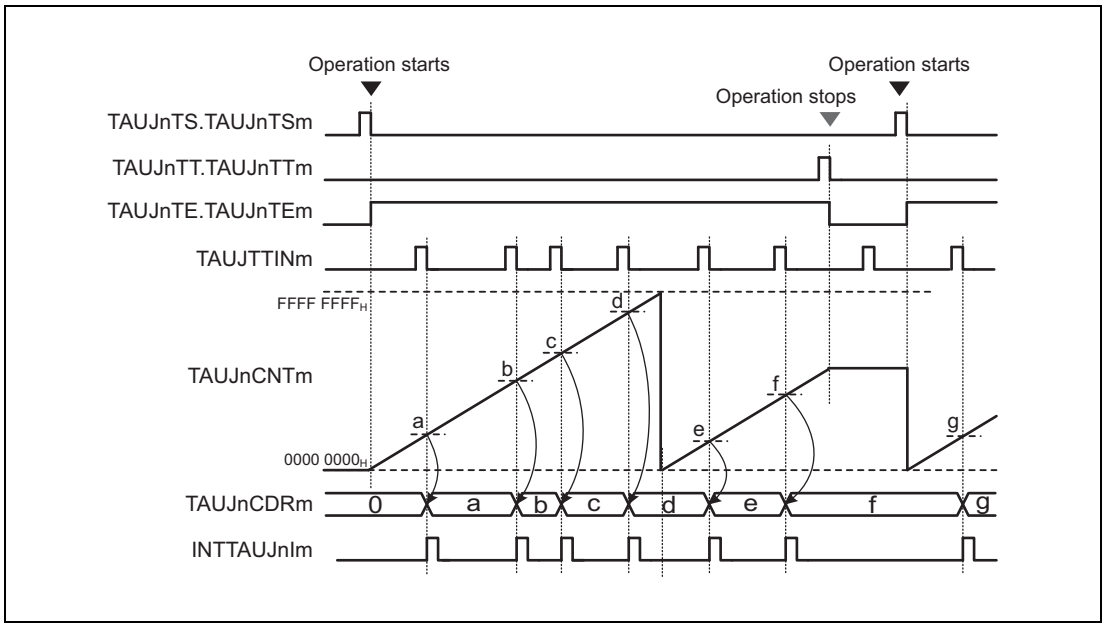



**Figure 26.38 Operation Stop and Restart (TAUJnCMORm.TAUJnMD0 = 0, TAUJnCMURm.TAUJnTIS[1:0] = $00_B$)**

- The counter can be stopped by setting TAUJnTT.TAUJnTTm to 1, which in turn sets TAUJnTE.TAUJnTEm to 0.
- TAUJnCNTm stops and the current value is retained.
- If the counter is stopped, valid TAUJTTINm input edges are ignored.
- The counter can be restarted by setting TAUJnTS.TAUJnTSm to 1. TAUJnCNTm restarts to count from 0000 $0000_H$.

## 26.12.6 TAUJTTINm Input Period Count Detection Function

### 26.12.6.1 Overview

#### Summary

This function measures the cumulative width of a TAUJTTINm input signal.

#### Prerequisites

TAUJTTOUTm is not used for this function.

#### Description

The counter is enabled by setting the channel trigger bit (TAUJnTS.TAUJnTSm) to 1. This in turn sets TAUJnTE.TAUJnTEm = 1, enabling count operation. The counter awaits a valid TAUJTTINm input edge.

When a valid TAUJTTINm input start edge is detected, the counter starts to count from 0000 $0000_H$.

When a valid TAUJTTINm input stop edge is detected, the current TAUJnCNTm value is loaded to TAUJnCDRm and an interrupt (INTTAUJnIm) is generated. The counter stops and retains its value until the next valid TAUJTTINm input start edge is detected.

When the next valid TAUJTTINm input start edge is detected, the counter restarts counting from the stop value.

When the counter reaches FFFF $FFFF_H$, the counter restarts from 0000 $0000_H$.

This function cannot be forcibly restarted.

**NOTE**

The input TAUJTTINm signal is sampled at the frequency of the operation clock, specified by the TAUJnCMORm.TAUJnCKS[1:0] bits.

#### Conditions

The valid start and stop edges are specified by the TAUJnCMURm.TAUJnTIS[1:0] bits.

- If TAUJnCMURm.TAUJnTIS[1:0] = $10_B$, the TAUJTTINm input low period is counted. The start trigger is a falling edge and the stop trigger is a rising edge.
- If TAUJnCMURm.TAUJnTIS[1:0] = $11_B$, the TAUJTTINm input high period is counted. The start trigger is a rising edge and the stop trigger is a falling edge.

### 26.12.6.2 Equations

Cumulative TAUJTTINm input width =
count clock cycle × (TAUJnCDRm capture value + 1)

### 26.12.6.3 Block Diagram and General Timing Diagram

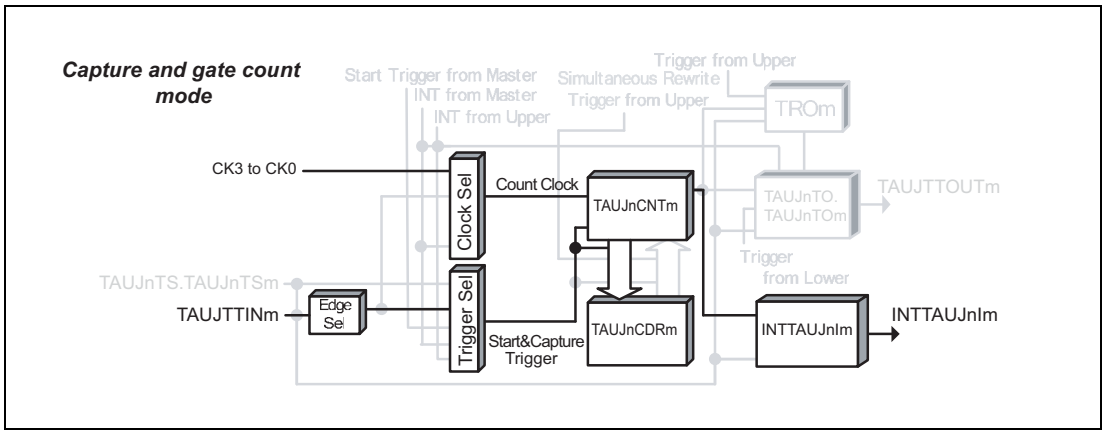



**Figure 26.39 Block Diagram for TAUJTTINm Input Period Count Detection Function**

The following settings apply to the general timing diagram.

- Rising and falling edge detection = high width measurement (TAUJnCMURm.TAUJnTIS[1:0] = $11_B$)

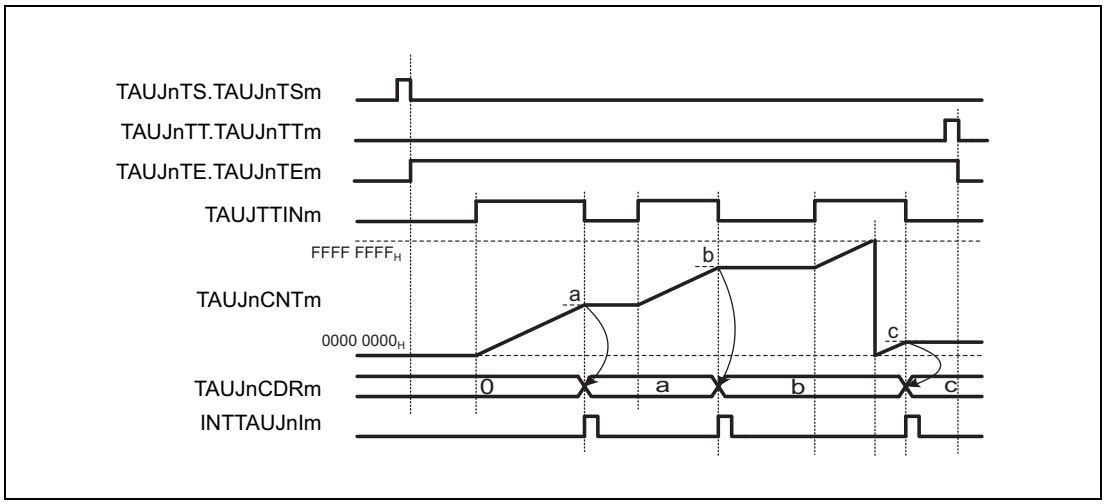



**Figure 26.40 General Timing Diagram for TAUJTTINm Input Period Count Detection Function**

### 26.12.6.4 Register Settings

#### (1) TAUJnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUJnCKS[1:0] | | TAUJnCCS[1:0] | | TAUJn MAS | TAUJnSTS[2:0] | | | TAUJnCOS[1:0] | | — | TAUJnMD[4:1] | | | | TAUJn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 26.62 Contents of the TAUJnCMORm Register for TAUJTTINm Input Period Count Detection Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUJnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output CK0<br>01: Prescaler output CK1<br>10: Prescaler output CK2<br>11: Prescaler output CK3 |
| 13, 12 | TAUJnCCS[1:0] | Write $00_B$. |
| 11 | TAUJnMAS | Write $0_B$. |
| 10 to 8 | TAUJnSTS[2:0] | Write $010_B$. |
| 7, 6 | TAUJnCOS[1:0] | Write $01_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUJnMD[4:1] | Write $1101_B$. |
| 0 | TAUJnMD0 | Write $0_B$. |

#### (2) TAUJnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUJnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 26.63 Contents of the TAUJnCMURm Register for TAUJTTINm Input Period Count Detection Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUJnTIS[1:0] | 10: Rising and falling edge detection (Low width measurement)<br>11: Rising and falling edge detection (High width measurement) |

#### (3) Channel output mode

TAUJnTOE.TAUJnTOEm is set to 0 because the channel output mode is not used with this function.

#### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUJnRDE and TAUJnRDM) cannot be used with the TAUJTTINm input period count detection function. Therefore, these registers must be set to 0.

**Table 26.64 Simultaneous Rewrite Settings for TAUJTTINm Input Period Count Detection Function**

| Bit Name | Setting |
|---|---|
| TAUJnRDE.TAUJnRDEm | 0: Disables simultaneous rewrite |
| TAUJnRDM.TAUJnRDMm | 0: When simultaneous rewrite is disabled (TAUJnRDE.TAUJnRDEm=0), set these bits to 0 |

### 26.12.6.5 Operating Procedure for TAUJTTINm Input Period Count Detection Function

**Table 26.65 Operating Procedure for TAUJTTINm Input Period Count Detection Function**

| | Operation | Status of TAUJn |
|---|---|---|
| Initial channel setting | Set the TAUJnCMORm and TAUJnCMURm registers as described in **Table 26.62, Contents of the TAUJnCMORm Register for TAUJTTINm Input Period Count Detection Function** and **Table 26.63, Contents of the TAUJnCMURm Register for TAUJTTINm Input Period Count Detection Function**.<br><br>The TAUJnCDRm register functions as a capture register. | Channel operation is stopped. |
| Start operation | Set TAUJTS.TAUJnTSm to 1.<br>TAUJnTS.TAUJnTSm is a trigger bit, so it is automatically cleared to 0. | TAUJnTE.TAUJnTEm is set to 1 and TAUJnCNTm waits for detection of the TAUJTTINm start edge. |
| During operation | TAUJTTINm edge detection<br><br>The TAUJnCDRm, TAUJnCNTm, and TAUJnCSRm registers can be read at any time. | When a TAUJTTINm start edge (rising edge for high width measurement, falling edge for low width measurement) is detected, TAUJnCNTm starts to count up from the stop value.<br>When TAUJnCNTm detects a stop edge (falling edge for high width measurement, rising edge for low width measurement), it transfers the value to TAUJnCDRm and INTTAUJnIm is generated.<br>Counting stops at the “value transferred to TAUJnCDRm + 1” and TAUJnCNTm waits for detection of the TAUJTTINm start edge.<br>When TAUJnCNTm reaches FFFF FFFF$_H$, the counter restarts from 0000 0000$_H$.<br>Afterwards, this procedure is repeated. |
| Stop operation | Set TAUJnTT.TAUJnTTm to 1.<br>TAUJnTT.TAUJnTTm is a trigger bit, so it is automatically cleared to 0. | TAUJnTE.TAUJnTEm is cleared to 0 and the counter stops.<br>TAUJnCNTm stops and retains its current value. |

Restart operation

### 26.12.6.6 Specific Timing Diagrams

#### (1) Operation Stop and Restart

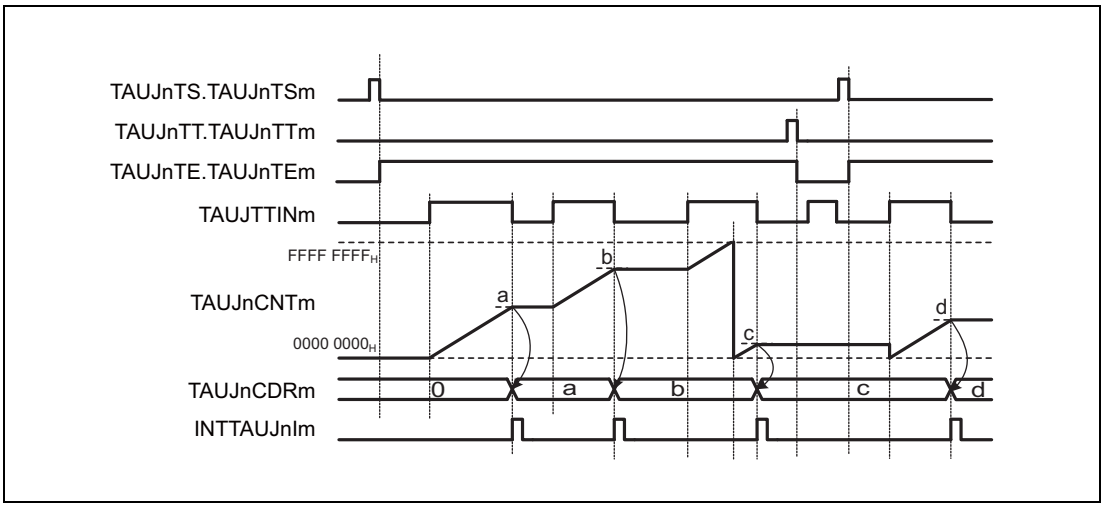



**Figure 26.41 Operation Stop and Restart (TAUJnCMURm.TAUJnTIS[1:0] = $11_B$)**

- The counter can be stopped by setting TAUJnTT.TAUJnTTm to 1, which in turn sets TAUJnTE.TAUJnTEm to 0.
- TAUJnCNTm stops and the current value is retained.
- If the counter is stopped, valid TAUJTTINm input edges are ignored.
- The counter can be restarted by setting TAUJnTS.TAUJnTSm to 1. TAUJnCNTm restarts to count from $0000\ 0000_H$.

## 26.12.7 Overflow Interrupt Output Function (during TAUJTTINm Width Measurement)

### 26.12.7.1 Overview

#### Summary

This function measures the width of an individual TAUJTTINm input signal. An interrupt is generated if the TAUJTTINm input width is longer than FFFF $FFFF_H$ + 1.

#### Prerequisites

- TAUJTTOUTm is not used for this function.
- The value of TAUJnCDRm must be set to FFFF $FFFF_H$.

#### Description

he counter is enabled by setting the channel trigger bit (TAUJnTS.TAUJnTSm) to 1. This in turn sets TAUJnTE.TAUJnTEm = 1, enabling count operation.

The counter starts when a valid TAUJTTINm input start edge is detected. FFFF $FFFF_H$ is loaded to TAUJnCNTm and the counter starts to count down.

When a valid stop edge is detected, the counter stops and retains the current value.

When the next TAUJTTINm input start edge is detected, TAUJnCNTm loads FFFF $FFFF_H$ and starts to count down.

If the counter reaches 0000 $0000_H$ before a stop edge is detected, an interrupt is generated.

#### Conditions

The valid start and stop edges are specified by the TAUJnCMURm.TAUJnTIS[1:0] bits.

- If TAUJnCMURm.TAUJnTIS[1:0] = $10_B$, the TAUJTTINm input low width is measured. The start trigger is a falling edge and the stop trigger is a rising edge.
- If TAUJnCMURm.TAUJnTIS[1:0] = $11_B$, the TAUJTTINm input high width is measured. The start trigger is a rising edge and the stop trigger is a falling edge.

**NOTE**

The counter cannot be restarted during operation.

### 26.12.7.2 Block Diagram and General Timing Diagram

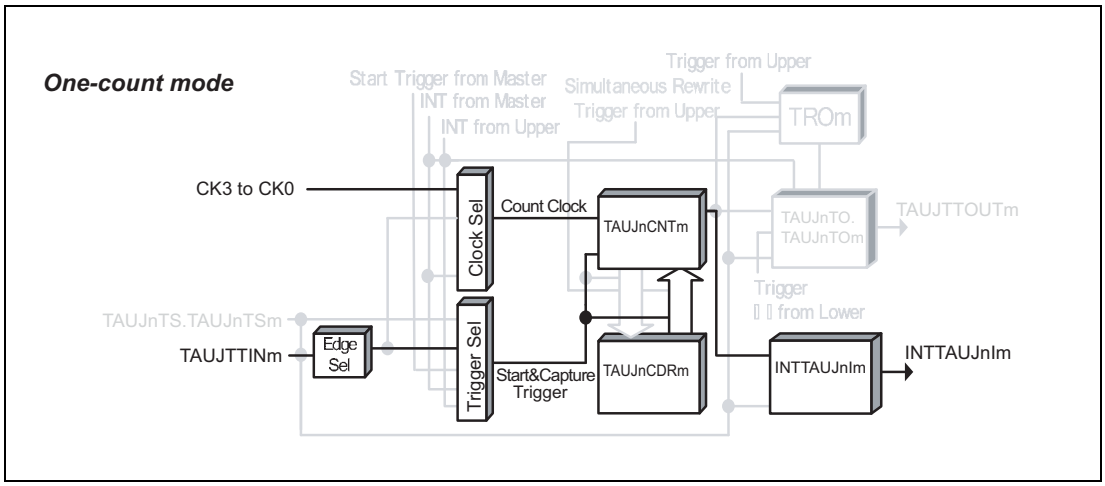



**Figure 26.42 Block Diagram for Overflow Interrupt Output Function (during TAUJTTINm Width Measurement)**

The following settings apply to the general timing diagram.

- Rising and falling edge detection = high width measurement (TAUJnCMURm.TAUJnTIS[1:0] = $11_B$)

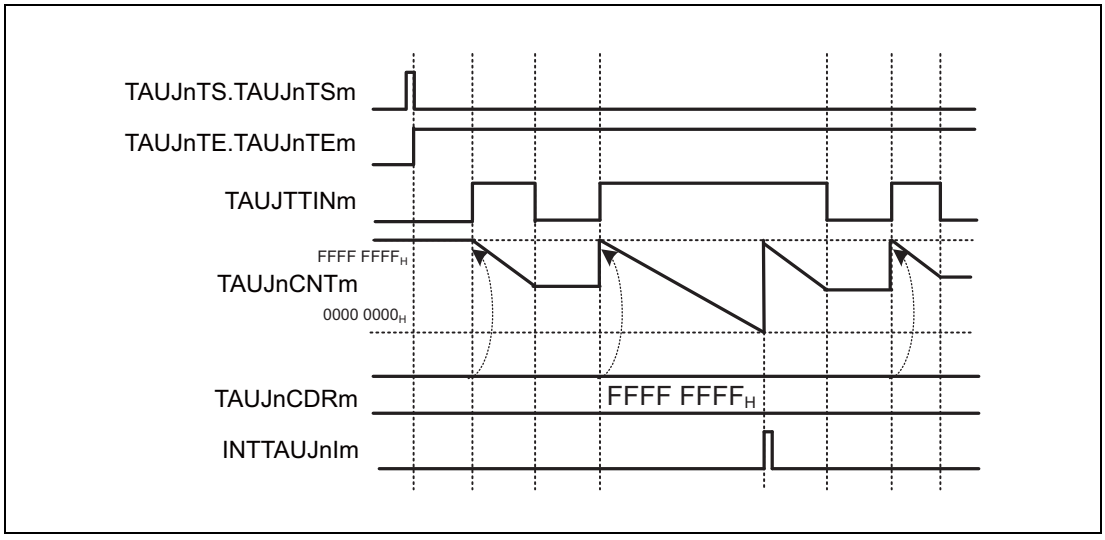



**Figure 26.43 General Timing Diagram for Overflow Interrupt Output Function (during TAUJTTINm Width Measurement)**

### 26.12.7.3 Register Settings

#### (1) TAUJnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUJnCKS[1:0] | | TAUJnCCS[1:0] | | TAUJn MAS | TAUJnSTS[2:0] | | | TAUJnCOS[1:0] | | — | TAUJnMD[4:1] | | | | TAUJn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 26.66 Contents of the TAUJnCMORm Register for Overflow Interrupt Output Function (during TAUJTTINm Width Measurement)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUJnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output = CK0<br>01: Prescaler output = CK1<br>10: Prescaler output = CK2<br>11: Prescaler output = CK3 |
| 13, 12 | TAUJnCCS[1:0] | Write $00_B$. |
| 11 | TAUJnMAS | Write $0_B$. |
| 10 to 8 | TAUJnSTS[2:0] | Write $010_B$. |
| 7, 6 | TAUJnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUJnMD[4:1] | Write $0100_B$. |
| 0 | TAUJnMD0 | Write $0_B$. |

#### (2) TAUJnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUJnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 26.67 Contents of the TAUJnCMURm Register for Overflow Interrupt Output Function (during TAUJTTINm Width Measurement)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUJnTIS[1:0] | 10: Rising and falling edge detection (Low width measurement)<br>11: Rising and falling edge detection (High width measurement) |

#### (3) Channel output mode

TAUJnTOE.TAUJnTOEm is set to 0 because the channel output mode is not used with this function.

#### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUJnRDE and TAUJnRDM) cannot be used with the overflow interrupt output function (during TAUJTTINm width measurement). Therefore, these registers must be set to 0.

**Table 26.68 Simultaneous Rewrite Settings for Overflow Interrupt Output Function (during TAUJTTINm Width Measurement)**

| Bit name | Setting |
|---|---|
| TAUJnRDE.TAUJnRDEm | 0: Disables simultaneous rewrite |
| TAUJnRDM.TAUJnRDMm | 0: When simultaneous rewrite is disabled (TAUJnRDE.TAUJnRDEm = 0), set these bits to 0 |

### 26.12.7.4 Operating Procedure for Overflow Interrupt Output Function (during TAUJTTINm Width Measurement)

**Table 26.69 Operating Procedure for Overflow Interrupt Output Function (during TAUJTTINm Width Measurement)**

| | Operation | Status of TAUJn |
|---|---|---|
| Initial channel setting | Set the TAUJnCMORm register and TAUJnCMURm registers as described in **Table 26.66, Contents of the TAUJnCMORm Register for Overflow Interrupt Output Function (during TAUJTTINm Width Measurement)** and **Table 26.67, Contents of the TAUJnCMURm Register for Overflow Interrupt Output Function (during TAUJTTINm Width Measurement)**.<br><br>Set the value of the TAUJnCDRm register to FFFF FFFF$_H$. | Channel operation is stopped. |
| Start operation | Set TAUJnTS.TAUJnTSm to 1.<br>TAUJnTS.TAUJnTSm is a trigger bit, so it is automatically cleared to 0.<br><br>Detection of TAUJTTINm start edge. | TAUJnTE.TAUJnTEm is set to 1 and TAUJnCNTm waits for detection of the start edge.<br><br>When a start edge is detected, TAUJnCNTm loads the TAUJnCDRm value (FFFF FFFF$_H$). |
| During operation | The TAUJnCNTm register can be read at any time. | TAUJnCNTm counts down. When the counter reaches 0000 0000$_H$:<br>• INTTAUJnIm is generated.<br>When a reverse edge of TAUJTTINm is detected during count operation:<br>• TAUJnCNTm stops counting and waits for a trigger.<br>Afterwards, this procedure is repeated. |
| Stop operation | Set TAUJnTT.TAUJnTTm to 1.<br>TAUJnTT.TAUJnTTm is a trigger bit, so it is automatically cleared to 0. | TAUJnTE.TAUJnTEm is cleared to 0 and the counter stops.<br>TAUJnCNTm stops and retains its current value. |

Restart operation

## 26.12.8 Overflow Interrupt Output Function (during TAUJTTINm Input Period Count Detection)

### 26.12.8.1 Overview

#### Summary

This function measures the cumulative width of a TAUJTTINm input signal. An interrupt is generated if the cumulative TAUJTTINm input width is longer than FFFF FFFF$_H$, and an overflow interrupt can be output.

#### Prerequisites

- TAUJTTOUTm is not used for this function.
- The value of TAUJnCDRm must be set to FFFF FFFF$_H$.

#### Description

The counter is enabled by setting the channel trigger bit (TAUJnTS.TAUJnTSm) to 1. This in turn sets TAUJnTE.TAUJnTEm = 1, enabling count operation.

The counter starts when a valid TAUJTTINm input start edge is detected. FFFF FFFF$_H$ is loaded to TAUJnCNTm and the counter starts to count down.

When a valid stop edge is detected, the counter stops and retains the current value. The counter awaits the next TAUJTTINm input start edge and then continues to count down from the current value.

When the counter reaches 0000 0000$_H$ an interrupt is generated. FFFF FFFF$_H$ is loaded to TAUJnCNTm and the counter continues to count down until a TAUJTTINm input stop edge is detected.

#### Conditions

The valid start and stop edges are specified by the TAUJnCMURm.TAUJnTIS[1:0] bits.

- If TAUJnCMURm.TAUJnTIS[1:0] = 10$_B$, the TAUJTTINm input low period is counted. The start trigger is a falling edge and the stop trigger is a rising edge.
- If TAUJnCMURm.TAUJnTIS[1:0] = 11$_B$, the TAUJTTINm input high period is counted. The start trigger is a rising edge and the stop trigger is a falling edge.

**NOTE**

The counter cannot be restarted during operation.

### 26.12.8.2 Block Diagram and General Timing Diagram

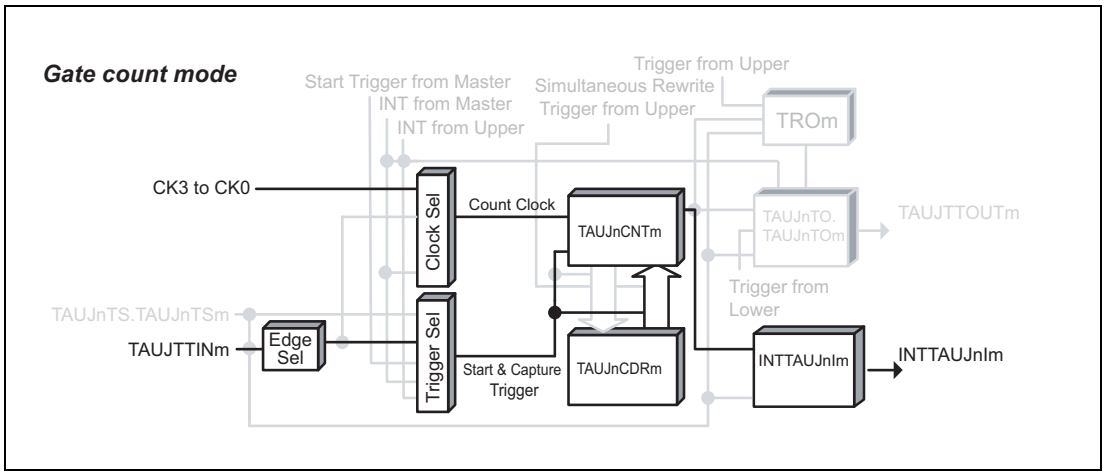



**Figure 26.44 Block Diagram for Overflow Interrupt Output Function (during TAUJTTINm Input Period Count Detection)**

The following settings apply to the general timing diagram.

- Rising and falling edge detection = high width measurement (TAUJnCMURm.TAUJnTIS[1:0] = $11_B$)

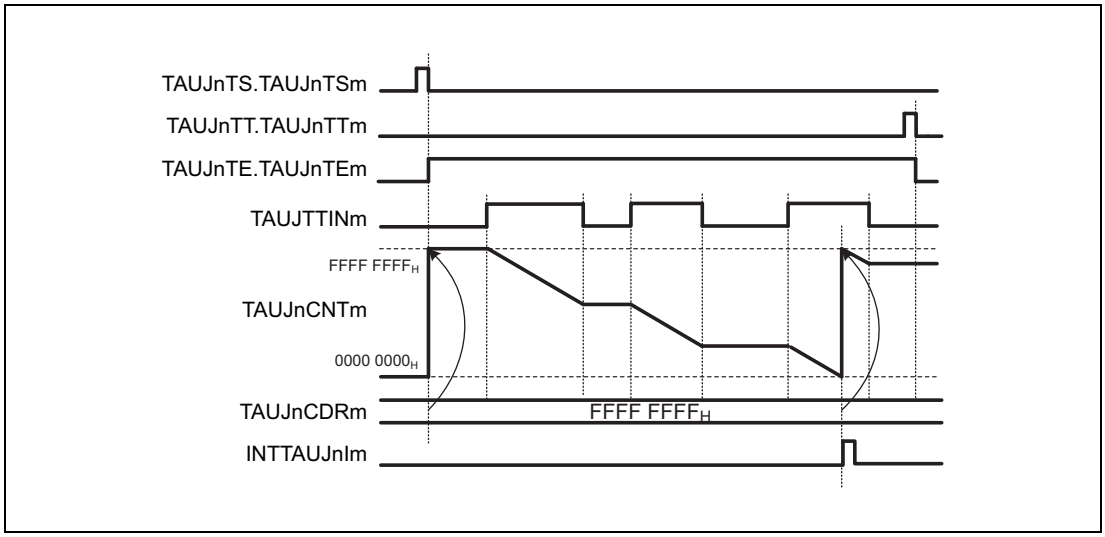



**Figure 26.45 General Timing Diagram For Overflow Interrupt Output Function (during TAUJTTINm Input Period Count Detection)**

### 26.12.8.3 Register Settings

#### (1) TAUJnCMORm

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUJnCKS[1:0] | | TAUJnCCS[1:0] | | TAUJn MAS | TAUJnSTS[2:0] | | | TAUJnCOS[1:0] | | — | TAUJnMD[4:1] | | | | TAUJn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 26.70 Contents of the TAUJnCMORm Register for Overflow Interrupt Output Function (during TAUJTTINm Input Period Count Detection)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUJnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output CK0<br>01: Prescaler output CK1<br>10: Prescaler output CK2<br>11: Prescaler output CK3 |
| 13, 12 | TAUJnCCS[1:0] | Write $00_B$. |
| 11 | TAUJnMAS | Write $0_B$. |
| 10 to 8 | TAUJnSTS[2:0] | Write $010_B$. |
| 7, 6 | TAUJnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUJnMD[4:1] | Write $1100_B$. |
| 0 | TAUJnMD0 | Write $0_B$. |

#### (2) TAUJnCMURm

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUJnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 26.71 Contents of the TAUJnCMURm Register for Overflow Interrupt Output Function (during TAUJTTINm Input Period Count Detection)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUJnTIS[1:0] | 10: Rising and falling edge detection (Low width measurement)<br>11: Rising and falling edge detection (High width measurement) |

#### (3) Channel output mode

TAUJnTOE.TAUJnTOEm is set to 0 because the channel output mode is not used with this function.

### (4) Simultaneous rewrite

The simultaneous rewrite registers (TAUJnRDE and TAUJnRDM) cannot be used with the Overflow Interrupt Output Function (During TAUJTTINm Input Period Count Detection). Therefore, these registers must be set to 0.

**Table 26.72 Simultaneous Rewrite Settings for Overflow Interrupt Output Function (during TAUJTTINm Input Period Count Detection)**

| Bit Name | Setting |
|---|---|
| TAUJnRDE.TAUJnRDEm | 0: Disables simultaneous rewrite. |
| TAUJnRDM.TAUJnRDMm | 0: When simultaneous rewrite is disabled (TAUJnRDE.TAUJnRDEm = 0), set these bits to 0. |

## 26.12.8.4 Operating Procedure for Overflow Interrupt Output Function (during TAUJTTINm Input Period Count Detection)

**Table 26.73 Operating Procedure for Overflow Interrupt Output Function (during TAUJTTINm Input Period Count Detection)**

| | Operation | Status of TAUJn |
|---|---|---|
| Initial channel setting | Set the TAUJnCMORm and TAUJnCMURm registers as described in **Table 26.70, Contents of the TAUJnCMORm Register for Overflow Interrupt Output Function (during TAUJTTINm Input Period Count Detection)** and **Table 26.71, Contents of the TAUJnCMURm Register for Overflow Interrupt Output Function (during TAUJTTINm Input Period Count Detection)**.<br><br>Set the value of the TAUJnCDRm register to FFFF FFFF$_H$. | Channel operation is stopped. |
| Start operation | Set TAUJnTS.TAUJnTSm to 1.<br>TAUJnTS.TAUJnTSm is a trigger bit, so it is automatically cleared to 0.<br><br>Detection of TAUJTTINm start edge. | TAUJnTE.TAUJnTEm is set to 1 and TAUJnCNTm waits for detection of the start edge.<br><br>When a start edge is detected, the value of TAUJnCDRm (FFFF FFFF$_H$) is loaded to TAUJnCNTm. |
| During operation | The TAUJnCNTm register can be read at all times | TAUJnCNTm counts down. When the counter reaches 0000 0000$_H$:<br>• INTTAUJnIm is generated.<br>• TAUJnCNTm loads the TAUJnCDRm value (FFFF FFFF$_H$) and continues to count down.<br>When a reverse edge of TAUJTTINm is detected during count operation:<br>• TAUJnCNTm stops and retains the stop value.<br>When a TAUJTTINm valid edge is detected while the counter is stopped:<br>• TAUJnCNTm counts down from the stop value.<br>Afterwards, this procedure is repeated. |
| Stop operation | Set TAUJnTT.TAUJnTTm to 1.<br>TAUJnTT.TAUJnTTm is a trigger bit, so it is automatically cleared to 0. | TAUJnTE.TAUJnTEm is cleared to 0 and the counter stops.<br>TAUJnCNTm stops and retains its current value. |

Restart operation

# 26.13 Synchronous Channel Operation Functions

This section lists all the synchronous channel operation functions provided by the TAUJ. For a general overview of synchronous channel operation, see **Section 26.2, Overview**.

## 26.13.1 PWM Output Function

### 26.13.1.1 Overview

#### Summary

This function generates multiple PWM outputs by using a master and multiple slave channels. It enables the pulse cycle (frequency) and the duty of the TAUJTTOUTm to be set. The pulse cycle is set in the master channel. The duty is set in the slave channel.

#### Prerequisites

- Two channels
- The operation mode for the master channel should be set to the interval timer mode. (See **Table 26.74, Contents of the TAUJnCMORm Register for the Master Channel of the PWM Output Function**.)
- The operation mode for the slave channel should be set to the one-count mode. (See **Table 26.77, Contents of the TAUJnCMORm Register for the Slave Channel of the PWM Output Function**.)
- TAUJTTOUTm is not used for the master channel of this function.
- The channel output mode for the slave channels should be set to synchronous channel output mode 1. (See **26.7, Channel Output Modes**.)

#### Description

The counters are enabled by setting the channel trigger bits (TAUJnTS.TAUJnTSm) to 1. This in turn sets TAUJnTE.TAUJnTEm = 1, enabling count operation. The current value of TAUJnCDRm is loaded to TAUJnCNTm and the counters start to count down from these values. INTTAUJnIm is generated on the master channel and TAUJTTOUTm (slave) is set or reset to realize the PWM output.

- Master channel:

  When the counter of the master channel reaches 0000 $0000_H$ and pulse cycle time has elapsed, INTTAUJnIm is generated. The TAUJnCDRm value is loaded to TAUJnCNTm, and the counter counts down.

- Slave channel(s):

  INTTAUJnIm generated on the master channel triggers the counter of the slave channel(s). The current value of TAUJnCDRm (slave) is loaded to TAUJnCNTm (slave) and the counter starts to count down from this value. The TAUJTTOUTm signal is set to the active level.
  When the counter reaches 0000 $0000_H$, i.e. duty time has elapsed, INTTAUJnIm is generated and the TAUJTTOUTm signal is set to the inactive level. The counter returns to FFFF $FFFF_H$ and awaits the next INTTAUJnIm of the master channel, and thus the start of the next pulse cycle.

The counter can be stopped by setting TAUJnTT.TAUJnTTm to 1 for the master and slave channel(s), which in turn sets TAUJnTE.TAUJnTEm to 0. TAUJnCNTm and TAUJTTOUTm of master and slave

channel(s) stop but retain their values. The counters can be restarted by setting TAUJnTS.TAUJnTSm to 1.

**Conditions**

Simultaneous rewrite can be used with this function. Please see **Section 26.6, Simultaneous Rewrite**.

### 26.13.1.2 Equations

Pulse cycle = (TAUJnCDRm (master) + 1) × count clock cycle

Duty cycle [%] = (TAUJnCDRm (slave)/(TAUJnCDRm (master) +1 )) × 100

– Duty cycle = 0%
  TAUJnCDRm (slave) = $0000\ 0000_H$
– Duty cycle = 100%
  TAUJnCDRm (slave) ≥ TAUJnCDRm (master) + 1

### 26.13.1.3 Block Diagram and General Timing Diagram

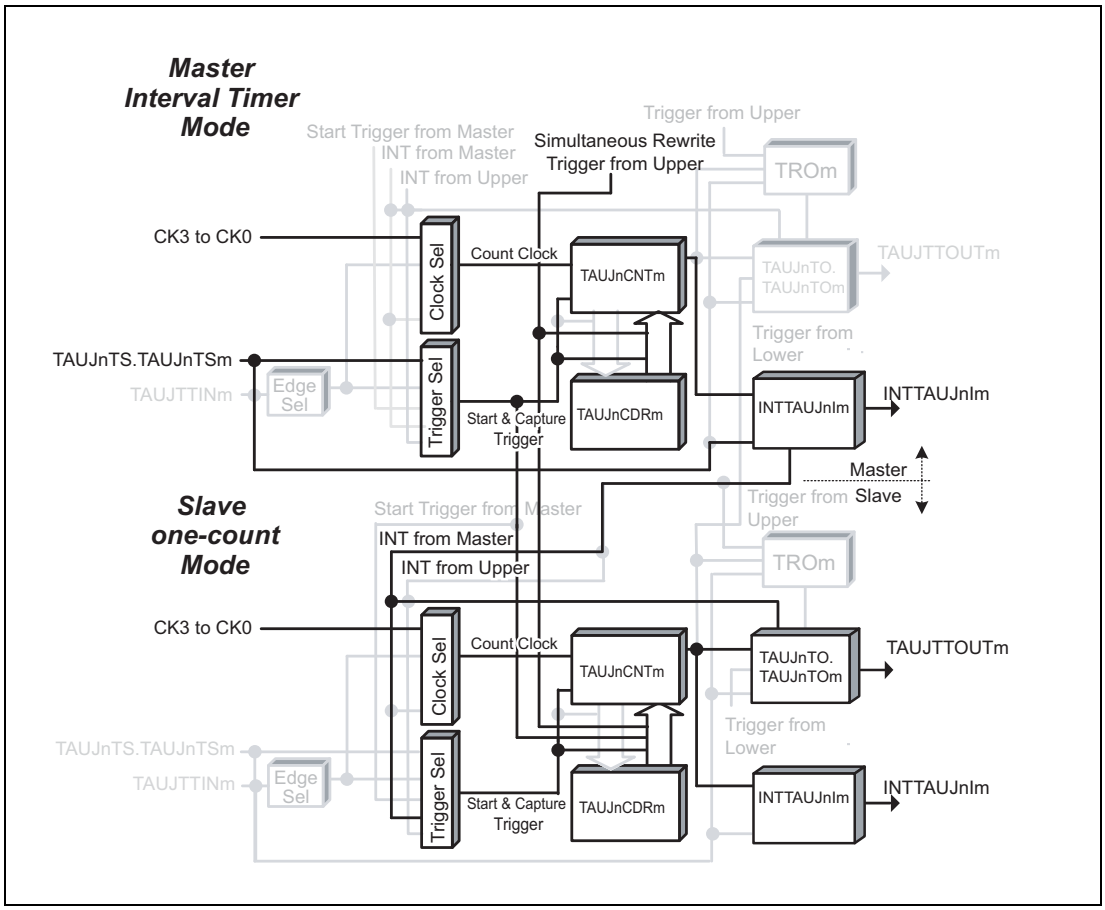



**Figure 26.46 Block Diagram for PWM Output Function**

The following settings apply to the general timing diagram.

- Slave channel: Positive logic (TAUJnTOL.TAUJnTOLm = 0)

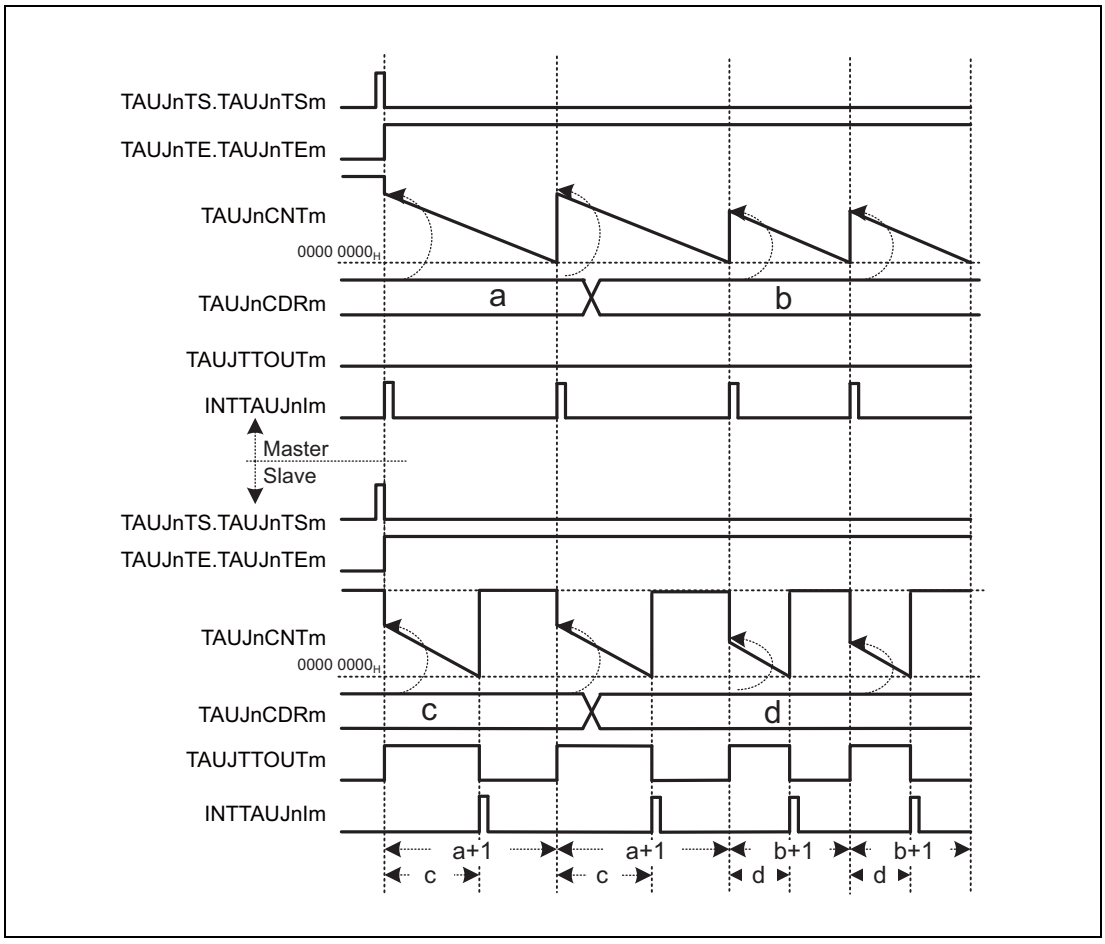



**Figure 26.47 General Timing Diagram for PWM Output Function**

**NOTE**

- The interval between the starting to count and an interrupt being generated is the value of corresponding TAUJnCDRm + 1.
- TAUJTTOUTm of the slave channel will rise with a delay of one count clock after the rising of INTTAUJnIm of the master channel.

### 26.13.1.4 Register Settings for the Master Channel

#### (1) TAUJnCMORm for the master channel

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUJnCKS[1:0] | | TAUJnCCS[1:0] | | TAUJn MAS | TAUJnSTS[2:0] | | | TAUJnCOS[1:0] | | — | TAUJnMD[4:1] | | | | TAUJn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 26.74 Contents of the TAUJnCMORm Register for the Master Channel of the PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUJnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output CK0<br>01: Prescaler output CK1<br>10: Prescaler output CK2<br>11: Prescaler output CK3<br>The value of the TAUJnCKS[1:0] bits of the master and slave channel(s) must be identical. |
| 13, 12 | TAUJnCCS[1:0] | Write $00_B$. |
| 11 | TAUJnMAS | Write $1_B$. |
| 10 to 8 | TAUJnSTS[2:0] | Write $000_B$. |
| 7, 6 | TAUJnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUJnMD[4:1] | Write $0000_B$. |
| 0 | TAUJnMD0 | Write $1_B$. |

#### (2) TAUJnCMURm for the master channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUJnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 26.75 Contents of the TAUJnCMURm Register for the Master Channel of the PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUJnTIS[1:0] | 00: Not used, so set to 00. |

#### (3) Channel output mode for the master channel

The channel output mode is not used by this function.

### (4) Simultaneous rewrite for the master channel

The simultaneous rewrite settings of the master and slave channels must be identical.

**Table 26.76 Simultaneous Rewrite Settings for the Master Channel of the PWM Output Function**

| Bit name | Setting |
|---|---|
| TAUJnRDE.TAUJnRDEm | 1: Enables simultaneous rewrite. |
| TAUJnRDM.TAUJnRDMm | 0: The simultaneous rewrite trigger signal is generated when the master channel starts counting. |

### 26.13.1.5 Register Settings for the Slave Channel(s)

#### (1) TAUJnCMORm for the slave channel(s)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAUJnCKS[1:0] | | TAUJnCCS[1:0] | | TAUJn MAS | TAUJnSTS[2:0] | | | TAUJnCOS[1:0] | | — | TAUJnMD[4:1] | | | | TAUJn MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

**Table 26.77 Contents of the TAUJnCMORm Register for the Slave Channel of the PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | TAUJnCKS[1:0] | Operation Clock Selection<br>00: Prescaler output CK0<br>01: Prescaler output CK1<br>10: Prescaler output CK2<br>11: Prescaler output CK3<br>The value of the TAUJnCKS[1:0] bits of the master and slave channel(s) must be identical. |
| 13, 12 | TAUJnCCS[1:0] | Write $00_B$. |
| 11 | TAUJnMAS | Write $0_B$. |
| 10 to 8 | TAUJnSTS[2:0] | Write $100_B$. |
| 7, 6 | TAUJnCOS[1:0] | Write $00_B$. |
| 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 1 | TAUJnMD[4:1] | Write $0100_B$. |
| 0 | TAUJnMD0 | Write $1_B$. |

#### (2) TAUJnCMURm for the slave channel(s)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TAUJnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 26.78 Contents of the TAUJnCMURm Register for the Slave Channel of the PWM Output Function**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | TAUJnTIS[1:0] | 00: Not used, so set to 00. |

### (3) Channel output mode for the slave channel(s)

**Table 26.79 Control Bit Settings for Synchronous Channel Output Mode 1**

| Bit Name | Setting |
|---|---|
| TAUJnTOE.TAUJnTOEm | Write $1_B$. |
| TAUJnTOM.TAUJnTOMm | Write $1_B$. |
| TAUJnTOC.TAUJnTOCm | Write $0_B$. |
| TAUJnTOL.TAUJnTOLm | 0: Positive logic<br>1: Negative logic |

### (4) Simultaneous rewrite for the slave channel(s)

The simultaneous rewrite settings of the master and slave channels must be identical.

**Table 26.80 Simultaneous Rewrite Settings for the Slave Channel of the PWM Output Function**

| Bit name | Setting |
|---|---|
| TAUJnRDE.TAUJnRDEm | 1: Enables simultaneous rewrite. |
| TAUJnRDM.TAUJnRDMm | 0: The simultaneous rewrite trigger signal is generated when the master channel starts counting. |

### 26.13.1.6 Operating Procedure for PWM Output Function

Table 26.81 Operating Procedure for PWM Output Function

| | Operation | Status of TAUJn |
|---|---|---|
| Initial channel setting | Master channel: Set the TAUJnCMORm and TAUJnCMURm registers and the channel output mode as described in **Section 26.13.1.4, Register Settings for the Master Channel**.<br><br>Slave channel: Set the TAUJnCMORm and TAUJnCMURm registers and the channel output mode as described in **Section 26.13.1.5, Register Settings for the Slave Channel(s)**.<br><br>Set the values of the TAUJnCDRm registers of all channels. | Channel operation is stopped. |
| Start operation | Set TAUJnTS.TAUJnTSm of the master and slave channels to 1 simultaneously.<br>TAUJnTS.TAUJnTSm is a trigger bit, so it is automatically cleared to 0. | TAUJnTE.TAUJnTEm (master and slave channels) is set to 1 and the counters of the master and slave channels start.<br>INTTAUJnIm is generated on the master channel and TAUJTTOUTm (slave) is set. |
| During operation | TAUJnCDRm can be changed at any time.<br>TAUJnCNTm and TAUJnRSF.TAUJnRSFm can be read at any time.<br><br>TAUJnRDT.TAUJnRDTm can be changed during operation. | TAUJnCNTm of the master channel loads TAUJnCDRm and counts down. When the counter reaches 0000 $0000_H$:<br>• INTTAUJnIm (master) is generated.<br>• TAUJnCNTm (master) loads the TAUJnCDRm value and continues count operation.<br>• TAUJnCNTm (slave) loads the TAUJnCDRm value and counts down.<br>• TAUJTTOUTm (slave) is set to the active level.<br>When TAUJnCNTm (slave) reaches 0000 $0000_H$:<br>• INTTAUJnIm (slave) is generated.<br>• TAUJTTOUTm (slave) is set to the inactive level. |
| Stop operation | Set TAUJnTT.TAUJnTTm of the master and slave channels to 1 simultaneously.<br>TAUJnTT.TAUJnTTm is a trigger bit, so it is automatically cleared to 0. | TAUJnTE.TAUJnTEm is cleared to 0 and the counter stops.<br>TAUJnCNTm and TAUJTTOUTm stop and retain their current values. |

Restart operation

### 26.13.1.7 Specific Timing Diagrams

#### (1) Duty cycle = 0%

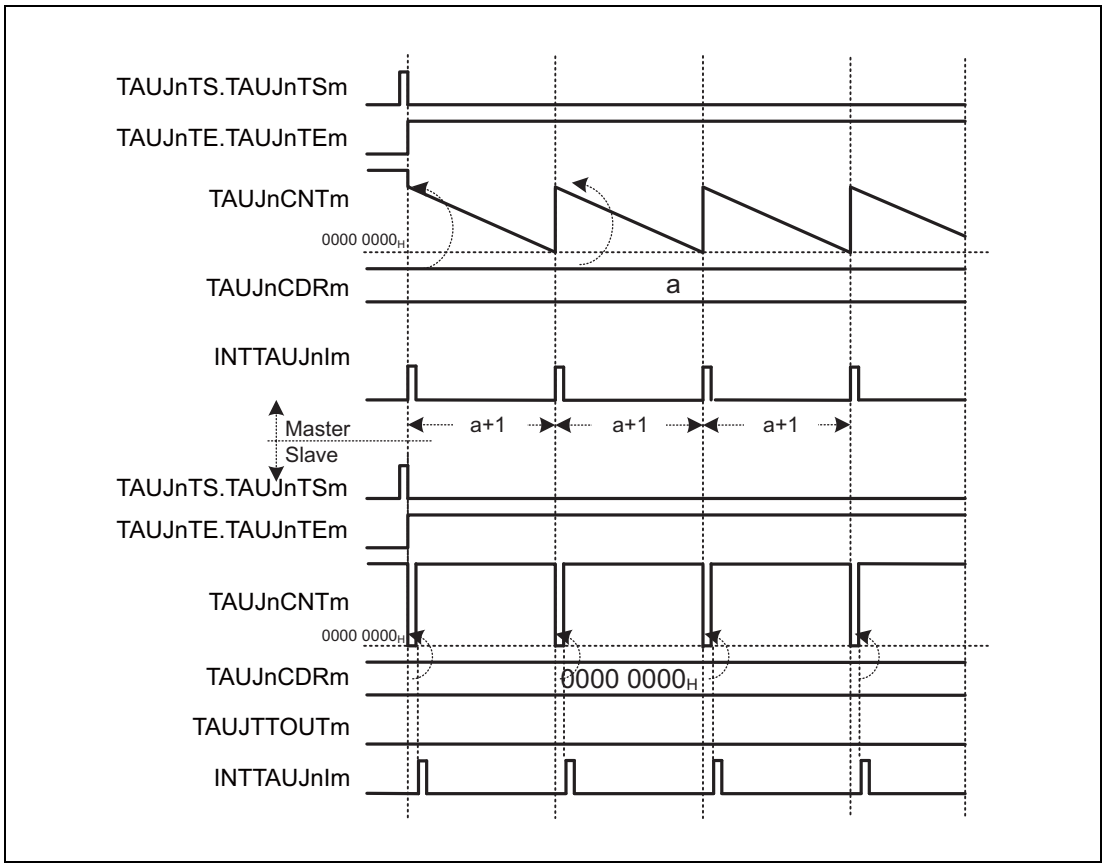



**Figure 26.48 TAUJnCDRm (slave) = 0000 0000$_H$, Positive Logic (TAUJnTOL.TAUJnTOLm (slave) = 0)**

- Every time the master channel generates an interrupt (INTTAUJnIm), 0000 0000$_H$ is loaded to TAUJnCNTm (slave). As a result, a slave channel interrupt (INTTAUJnIm) is generated at the same time and TAUJTTOUTm remains inactive.
- The value of TAUJnCDRm is loaded into TAUJnCNTm (slave) to generate an interrupt.

**(2) Duty cycle = 100%**

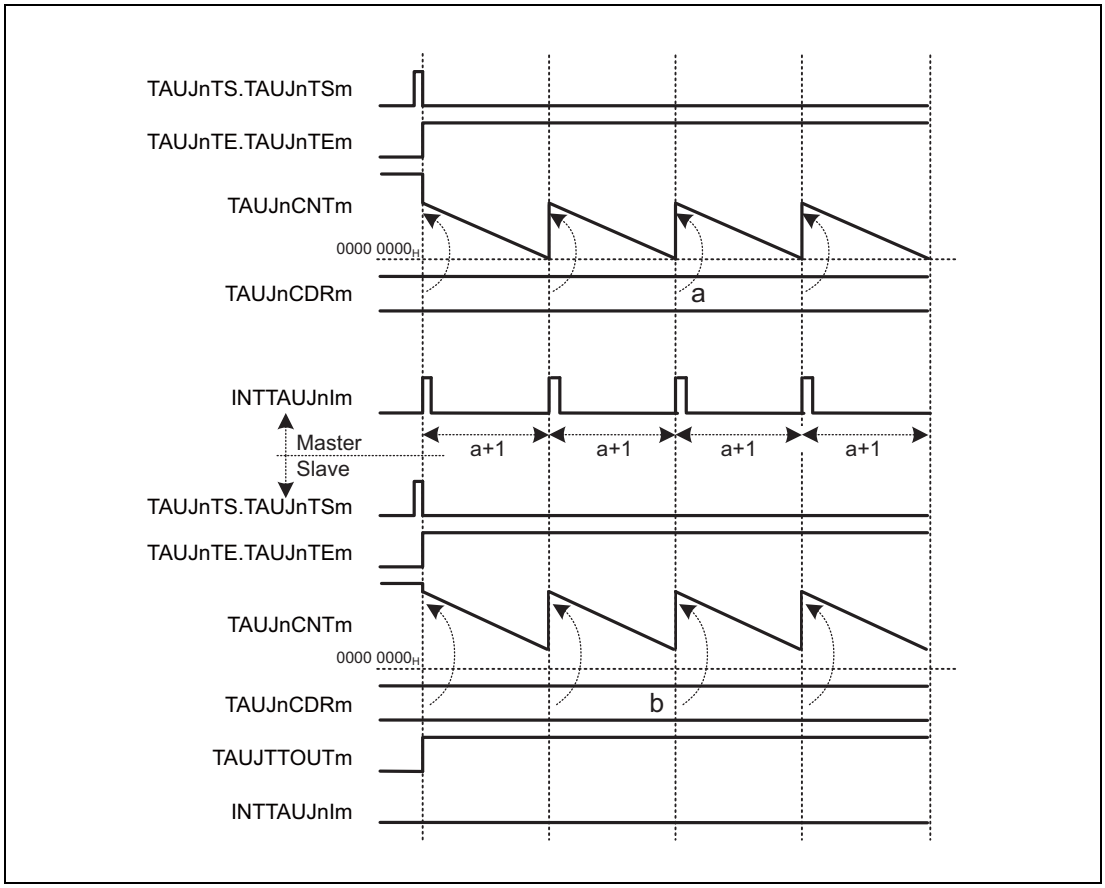



**Figure 26.49 TAUJnCDRm (slave) ≥ TAUJnCDRm (master) + 1, Positive Logic (TAUJnTOL.TAUJnTOLm (slave) = 0)**

If the TAUJnCDRm (slave) value is greater than the TAUJnCDRm (master) value, no interrupt occurs because the counter of the slave channel does not reach 0000 $0000_H$. TAUJTTOUTm remains active.

### (3) Operation stop and restart

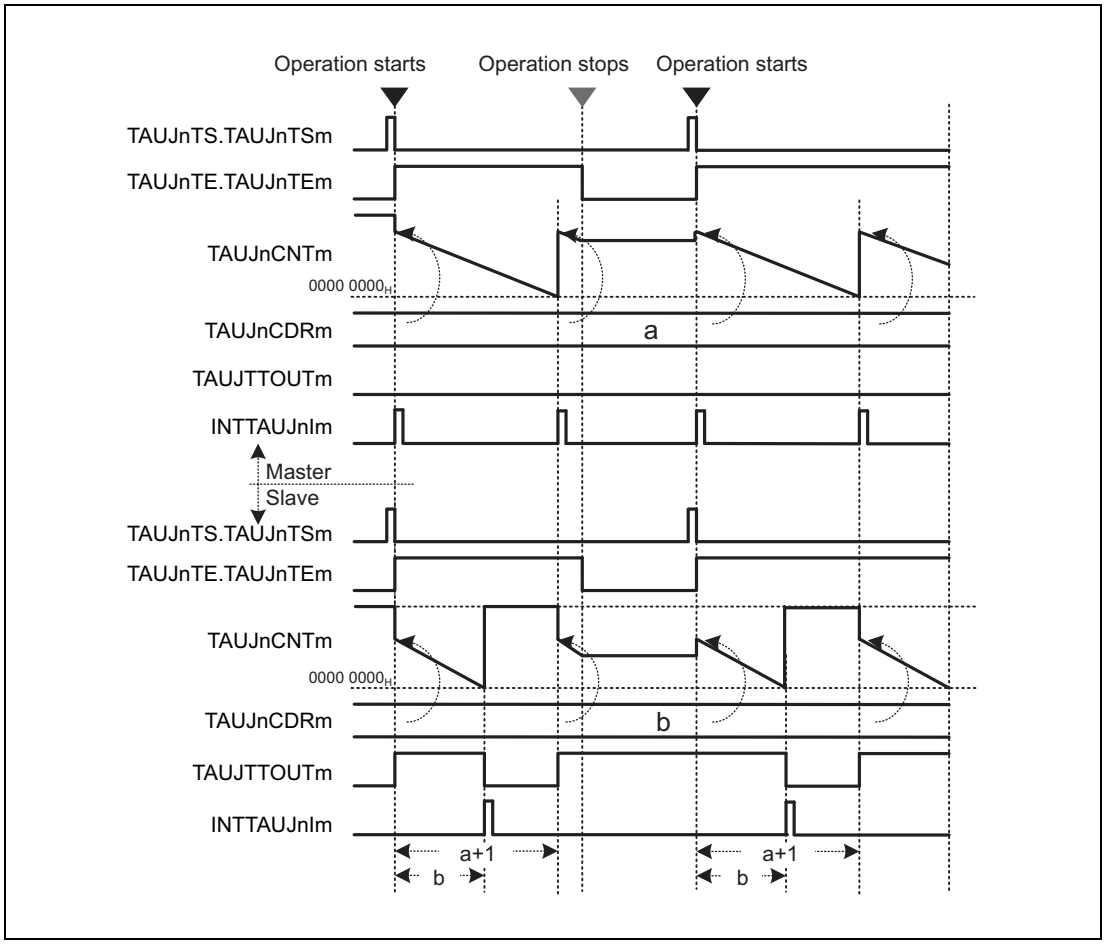



**Figure 26.50 Operation Stop and Restart, Positive Logic (TAUJnTOL.TAUJnTOLm (slave) = 0)**

- The counter can be stopped by setting TAUJnTT.TAUJnTTm of master and slave channels to 1. This sets TAUJnTE.TAUJnTEm to 0.
- TAUJnCNTm and TAUJTTOUTm of every channel stop and retain their current values. No interrupt occurs.
- The counter can be restarted by setting TAUJnTS.TAUJnTSm of master and slave channels to 1. The TAUJnCDRm value of master and slave channels is loaded into TAUJnCNTm. The counter starts to count down from this value.

# Section 27 Real-Time Clock (RTCA)

This section contains a generic description of the Real-Time Clock (RTCA).

The first part of this section describes all RH850/F1K specific properties, such as the number of units, register base addresses, etc. The remainder of the section describes the functions and registers of the RTCA.

## 27.1 Features of RH850/F1K RTCA

### 27.1.1 Number of Units and Channels

This microcontroller has the following number of RTCA units.

Each RTCA unit has one channel RTCA. “Number of channels” is used with the same meaning as “number of units” in this section.

**Table 27.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of Units | — | 1 | |
| Name | — | RTCAn (n = 0) | |

**Table 27.2 Index**

| Index | Description |
|---|---|
| n | Throughout this section, the individual RTCA units are identified by the index “n” (n = 0); for example, RTCAnCTL0 is the RTCAn control register 0. |

### 27.1.2 Register Base Address

RTCAn base addresses are listed in the following table.

RTCAn register addresses are given as offsets from the base addresses.

**Table 27.3 Register Base Address**

| Base Address Name | Base Address |
|---|---|
| <RTCA0_base> | FFE7 $8000_H$ |

### 27.1.3 Clock Supply

The RTCAn clock supply is shown in the following table.

**Table 27.4 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| RTCA0 | RTCATCKI | CKSCLK_ARTCA | Macro clock |
| | PCLK | CPUCLK2 | Module clock |
| | Register access clock | CPUCLK2 | Bus clock |

### 27.1.4 Interrupt Requests

RTCAn interrupt requests are listed in the following table.

**Table 27.5 Interrupt Requests**

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **RTCA0** | | | |
| RTCATINT1S | 1-second interval interrupt | 209 | — |
| RTCATINTAL | Alarm interrupt | 210 | — |
| RTCATINTR | Fixed interval interrupt | 211 | — |

### 27.1.5 Reset Sources

RTCAn reset sources are listed in the following table. RTCAn is initialized by these reset sources.

**Table 27.6 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| RTCA0 | Power-up reset (PURES) |

### 27.1.6 External Input/Output Signals

External input/output signals of RTCAn are listed below.

**Table 27.7 External Input/Output Signals**

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **RTCA0** | | |
| RTCAT1HZ | 1-Hz pulse output | RTCA0OUT*1 |

Note 1. RTCA0OUT is connected to TAUJ0. For details, see **Section 26, Timer Array Unit J (TAUJ)**.

**CAUTION**

**When port P8_6 is used as RTCA0OUT, port P8_6 pin outputs a low-level RESETOUT signal while a reset is asserted and continues to output a low level after the reset is deasserted. For details, see Section 2.11.1.1, P8_6: RESETOUT.**

## 27.2 Overview

### 27.2.1 Functional Overview

The Real-Time Clock (RTCA) has the following features:

- Count clock selection from 32 kHz to 4.194304 MHz
- Counters for years, months, day of the month, day of the week, hours, minutes, seconds, and a sub-counter. The calendar covers 99 years. Leap years are handled by hardware automatically.
- One Hz pulse output function
- Fixed interval interrupt function
- Alarm interrupt function
- Clock error correction function if a 32.768-kHz count clock is used

### 27.2.2 Block Diagram

The block diagram shows the main components of the RTCA.

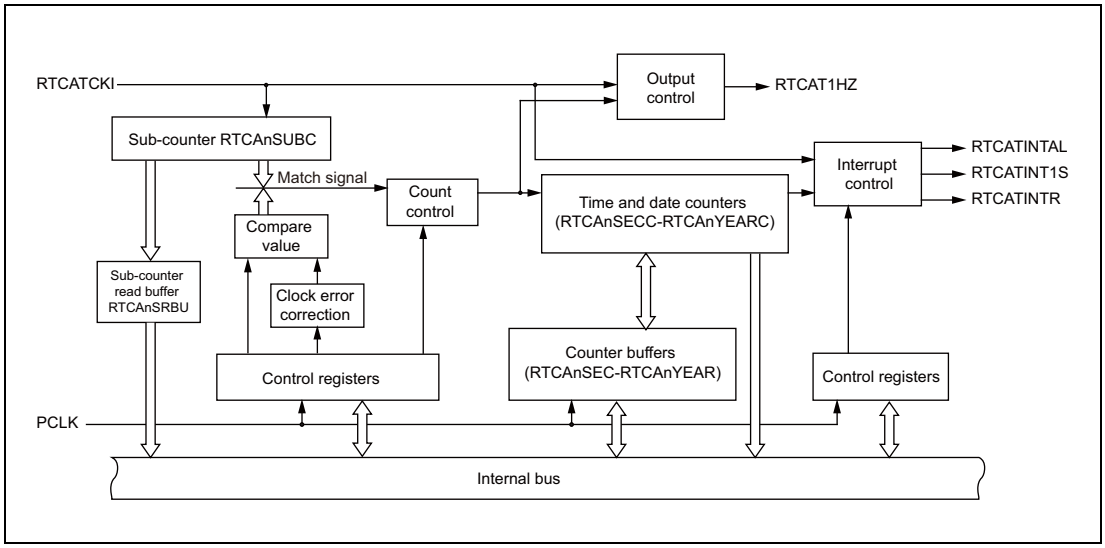



**Figure 27.1 Block Diagram of the RTCA**

### 27.2.3 Description of Blocks

The Real-Time Clock RTCA provides information about the present time and date and can generate wake-up signals (interrupts, alarms). This information is derived from the count clock RTCATCKI.

#### Sub-counter

RTCATCKI is the input to the sub-counter RTCAnSUBC. The sub-counter counts up from 0 until it reaches the compare value. The compare value is always defined as the frequency of RTCATCKI – 1 (in Hz). Thus, the sub-counter overflows after one second. It is then reset to 0 and triggers the seconds counter RTCAnSECC (and, if specified, the interrupt RTCATINT1S).

The sub-counter can generate a fixed interval interrupt every 0.25 seconds, 0.5 seconds, or 1 second, and a 1-Hz output pulse.

#### Time and date counters

The counters for minutes, hours, day of the week, day of the month, months, and years also count up. They have their own overflow limits. If all the lower counters overflow, the upper counter counts up.

The overflow limit of the counter for the day of the month (RTCAnDAYC) depends on the present month (28, 30, or 31 days) and (in February) on the year counter RTCAnYEARC (years 0, 4, 8, 12, etc. are considered leap years).

The hours counter RTCAnHOURC can be switched between 12- and 24-hour formats.

The counters for seconds, minutes, hours, day of the month, and months can generate a fixed interval interrupt upon overflow (RTCATINTR).

The counters for minutes, hours, and day of the week can also generate an alarm interrupt (RTCATINTAL), e.g. every Tuesday and Thursday at 10:32.

#### Counter buffers

All counters can be read directly at any time. The clock signal used to access the read/write registers and the count clock are usually asynchronous. An overflow of the sub-counter during the read operation can make all read values obsolete. Therefore, reading the counters must be performed using a special procedure. For details, see **Section 27.5.3, Reading Clock Counters**.

For reasons of synchronization, the counters cannot be written directly.

For reading and writing, all counters are accompanied by buffer registers. The buffer registers provide a synchronized way for reading the counters and for setting time and date. When they are used, the operation of the sub-counter must first be suspended and then re-activated (see also **Section 27.5.3, Reading Clock Counters** and **Section 27.5.2, Updating Clock Counters**.

The RTCAnTIMEC and RTCAnCALC registers and their corresponding buffer registers can be used to check and set the time (hours, minutes and seconds) or the date (day of the week, day of the month, month, and year) with one read/write operation.

# 27.3 Registers

## 27.3.1 List of Registers

RTCA registers are listed in the following table.

<RTCAn_base> is defined in **Section 27.1.2, Register Base Address**.

**Table 27.8 List of Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| **Control registers** | | | |
| RTCAn | Control register 0 | RTCAnCTL0 | <RTCAn_base> + $00_H$ |
| | Control register 1 | RTCAnCTL1 | <RTCAn_base> + $04_H$ |
| | Control register 2 | RTCAnCTL2 | <RTCAn_base> + $08_H$ |
| **Sub-counter registers** | | | |
| RTCAn | Sub-count register | RTCAnSUBC | <RTCAn_base> + $0C_H$ |
| | Sub-count register read buffer | RTCAnSRBU | <RTCAn_base> + $10_H$ |
| | Clock error correction register | RTCAnSUBU | <RTCAn_base> + $38_H$ |
| | Sub-counter compare register | RTCAnSCMP | <RTCAn_base> + $3C_H$ |
| **Clock counter and buffer registers** | | | |
| RTCAn | Seconds count register | RTCAnSECC | <RTCAn_base> + $4C_H$ |
| | Seconds count buffer register | RTCAnSEC | <RTCAn_base> + $14_H$ |
| | Minute count register | RTCAnMINC | <RTCAn_base> + $50_H$ |
| | Minute count buffer register | RTCAnMIN | <RTCAn_base> + $18_H$ |
| | Hour count register | RTCAnHOURC | <RTCAn_base> + $54_H$ |
| | Hour count buffer register | RTCAnHOUR | <RTCAn_base> + $1C_H$ |
| | Day of the week count register | RTCAnWEEKC | <RTCAn_base> + $58_H$ |
| | Day of the week count buffer register | RTCAnWEEK | <RTCAn_base> + $20_H$ |
| | Day count register | RTCAnDAYC | <RTCAn_base> + $5C_H$ |
| | Day count buffer register | RTCAnDAY | <RTCAn_base> + $24_H$ |
| | Month count register | RTCAnMONC | <RTCAn_base> + $60_H$ |
| | Month count buffer register | RTCAnMONTH | <RTCAn_base> + $28_H$ |
| | Year count register | RTCAnYEARC | <RTCAn_base> + $64_H$ |
| | Year count buffer register | RTCAnYEAR | <RTCAn_base> + $2C_H$ |
| **Special counter and buffer registers** | | | |
| RTCAn | Time count register | RTCAnTIMEC | <RTCAn_base> + $68_H$ |
| | Time count buffer register | RTCAnTIME | <RTCAn_base> + $30_H$ |
| | Calendar count register | RTCAnCALC | <RTCAn_base> + $6C_H$ |
| | Calendar count buffer register | RTCAnCAL | <RTCAn_base> + $34_H$ |
| **Alarm time setting registers** | | | |
| RTCAn | Alarm minute setting register | RTCAnALM | <RTCAn_base> + $40_H$ |
| | Alarm hour setting register | RTCAnALH | <RTCAn_base> + $44_H$ |
| | Alarm day of the week setting register | RTCAnALW | <RTCAn_base> + $48_H$ |
| **Emulation register** | | | |
| RTCAn | Emulation register | RTCAnEMU | <RTCAn_base> + $74_H$ |

## 27.3.2 Details of RTCA Control Registers

### 27.3.2.1 RTCAnCTL0 — RTCA Control Register 0

This register controls the count operation of the sub-counter RTCAnSUBC, the format (12-hour/24-hour) of the hours counter RTCAnHOURC and the alarm hour setting register RTCAnALH, and the operation mode.

**Access:** This register can be read or written in 8-bit or 1-bit units.

**Address:** <RTCAn_base> + $00_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RTCAnCE | RTCAnCEST | RTCAnAMPM | RTCAnSLSB | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R/W | R/W | R | R | R | R |

**Table 27.9 RTCAnCTL0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RTCAnCE | Starts/stops the sub-counter RTCAnSUBC operation.<br>0: Stops the sub-counter operation.<br>All output pins and all status flags in control register RTCAnCTL2 are cleared.<br>1: Starts the sub-counter operation.<br>The sub-counter counts up. |
| 6 | RTCAnCEST | Indicates the operation enabled/stopped status of the sub-counter:<br>0: Operation stopped status<br>1: Operation enabled status<br>For details on how to use this status flag, see **Section 27.5.1, Initial Setting of the RTCA**. |
| 5 | RTCAnAMPM | Selects the format of the hours counter RTCAnHOURC and the alarm hour setting register RTCAnALH:<br>0: 12-hour format (1 to 12, am/pm)<br>1: 24-hour format (0 to 23, military time)<br>For details on the format, see **Table 27.21, 12- and 24-Hour Format**. |
| 4 | RTCAnSLSB | Selects the operation mode:<br>0: 32.768 kHz mode<br>1: Frequency selection mode<br>For details on the operation modes, see **Section 27.4, Operation**.<br>The operation mode must not be changed while sub-counter operation is enabled (RTCAnCTL0.RTCAnCEST = 1).<br>For details on the initialization of RTCAn, see **Section 27.5.1, Initial Setting of the RTCA**. |
| 3 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

### 27.3.2.2 RTCAnCTL1 — RTCA Control Register 1

This register controls the interrupt request generation and the 1-Hz pulse output.

**Access:** This register can be read or written in 8-bit or 1-bit units.

**Address:** <RTCAn_base> + $04_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | RTCAnEN1HZ | RTCAnENALM | RTCAnEN1S | RTCAnCT2 | RTCAnCT1 | RTCAnCT0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.10 RTCAnCTL1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 | RTCAnEN1HZ | Enables/stops 1-Hz pulse output (RTCAT1HZ):<br>0: RTCAT1HZ disabled (RTCAT1HZ is fixed to 0)<br>1: RTCAT1HZ enabled |
| 4 | RTCAnENALM | Enables/disables alarm interrupt request generation (RTCATINTAL):<br>0: RTCATINTAL disabled<br>1: RTCATINTAL enabled |
| 3 | RTCAnEN1S | Enables/disables 1-second interrupt request generation (RTCATINT1S):<br>0: RTCATINT1S disabled<br>1: RTCATINT1S enabled |
| 2 to 0 | RTCAnCT[2:0] | Specifies the fixed interval interrupt request (RTCATINTR) setting: |

| RTCAnCT[2:0] | RTCATINTR Interrupt Request Generation | |
|---|---|---|
| | Interval | Timing |
| 000 | No interrupt request generation | |
| 001 | Every 0.25 seconds | Every 0.25, 0.5, 0.75 and 1 second |
| 010 | Every 0.5 seconds | Every 0.5 and 1 second |
| 011 | Every second | Every 1 second |
| 100 | Every minute | Every 1 minute 00 seconds |
| 101 | Every hour | Every 1 hour 00 minutes 00 seconds |
| 110 | Every day | Every 1 day 00 hours 00 minutes 00 seconds (i.e., every midnight) |
| 111 | Every month | Every 1 month first day 00 hours 00 minutes 00 seconds (i.e., every first midnight of a month) |

If the settings of RTCAnCT[2:0] are changed while sub-counter operation is enabled (RTCAnCTL0.RTCAnCE = 1), a glitch may be output to RTCATINTR. Implement appropriate interrupt mask processing procedures.

### 27.3.2.3 RTCAnCTL2 — RTCA Control Register 2

This register contains status information and controls the data transfer from the sub-counter RTCAnSUBC to the dedicated sub-counter read buffer RTCAnSRBU and the operation setting of the clock counters (RTCAnSECC to RTCAnYEARC).

**Access:** This register can be read or written in 8-bit or 1-bit units.

**Address:** <RTCAn_base> + $08_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | RTCAnWUST | RTCAnWSST | RTCAnRSST | RTCAnRSUB | RTCAnWST | RTCAnWAIT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R | R/W |

**Table 27.11 RTCAnCTL2 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 | RTCAnWUST | Indicates whether RTCAnSUBU write operation has been completed:<br>0: RTCAnSUBU write completed<br>1: RTCAnSUBU write in progress<br>The write operation ends with the next sub-counter overflow.<br>While the sub-counter operation is enabled (RTCAnCTL0.RTCAnCE = 1) and if write operation to RTCAnSUBU is completed, this bit is set to 1.<br><br>See **Section 27.5.5, Writing to RTCAnSUBU**, for details. |
| 4 | RTCAnWSST | Indicates whether RTCAnSCMP write operation has been completed:<br>0: RTCAnSCMP write completed<br>1: RTCAnSCMP write in progress<br>The write operation ends with the next sub-counter overflow.<br>While the sub-counter operation is enabled (RTCAnCTL0.RTCAnCE = 1) and if write operation to RTCAnSCMP is completed, this bit is set to 1.<br><br>See **Section 27.5.6, Writing to RTCAnSCMP**, for details. |
| 3 | RTCAnRSST | Indicates whether the value of the sub-counter (RTCAnSUBC) has been transferred to the sub-count register read buffer (RTCAnSRBU):<br>0: Transfer in progress, or waiting for a transfer trigger<br>1: Transfer completed<br>This bit is cleared (transfer is triggered) by RTCAnRSUB=1.This bit is automatically set when the transfer is completed.<br><br>See **Section 27.5.4, Reading RTCAnSRBU**, for details. |
| 2 | RTCAnRSUB | Triggers transfer of the value of the sub-counter (RTCAnSUBC) to the dedicated read buffer (RTCAnSRBU) or clears the transfer state of the sub-counter:<br>0: Transfer status (RTCAnRSST) is cleared.<br>1: Transfer is triggered.<br>This bit is used to read the value of RTCAnSRBU when the sub-counter operation is enabled (RTCAnCTL0.RTCAnCE = 1). The value of RTCAnSUBC is synchronized with RTCATCKI and loaded to RTCAnSRBU.<br><br>For details, see **Section 27.5.4, Reading RTCAnSRBU**. |
| 1 | RTCAnWST | Indicates the status of all clock counters (RTCAnSECC to RTCAnYEARC):<br>0: All clock counters are running.<br>1: All clock counters are stopped<br>The sub-counter is still running.<br><br>The clock counters must be stopped before reading or writing clock counter values during sub-counter operation (RTCAnCTL0.RTCAnCE = 1). To stop the clock counters, set RTCAnWAIT = 1. |

**Table 27.11 RTCAnCTL2 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | RTCAnWAIT | Restarts/stops all clock counters (RTCAnSECC to RTCAnYEARC):<br>0: Restarts all clock counters either immediately or immediately after the clock counter write operation finishes.<br>1: Stops all clock counters temporarily. The sub-counter is still running.<br><br>The clock counters must be stopped before reading or writing counter buffers during sub-counter operation (RTCAnCTL0.RTCAnCE = 1).<br><br>**CAUTION**<br>Only one overflow can be held internally. When two overflows occur, the seconds counter is incremented only by one when it is restarted. Thus, the procedure must be completed within one second. |

## 27.3.3 Details of RTCA Sub-Counter Registers

### 27.3.3.1 RTCAnSUBC — RTCA Sub-Count Register

This counter counts the 1-second reference time. It operates using the count clock RTCATCKI.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <RTCAn_base> + $0C_H$

**Value after reset:** $0000\ 0000_H$

This register is initialized:

– When write operation is performed to the seconds count buffer register (RTCAnSEC) or to the time count buffer register (RTCAnTIME) and the value is reflected to the seconds count register.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | RTCAnSUBC[21:16] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RTCAnSUBC[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 27.12 RTCAnSUBC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 22 | Reserved | When read, the value after reset is returned. |
| 21 to 0 | RTCAnSUBC [21:0] | Sub-counter value<br>The sub-counter only operates while RTCAnCTL0.RTCAnCEST = 1. |

**NOTES**

1. This sub-counter operates with RTCATCKI while the read operation is clocked by PCLK. Reading this sub-counter during operation (RTCAnCTL0.RTCAnCEST = 1) is asynchronous to RTCATCKI and can lead to wrong results.
   Use the sub-count register read buffer (RTCAnSRBU) to read the sub-counter value during operation.
   For details, see **Section 27.5.4, Reading RTCAnSRBU**.
2. The count-operation of this sub-counter depends on the selected operation mode. See **Section 27.4, Operation**, for details.

### 27.3.3.2 RTCAnSRBU — RTCA Sub-Count Register Read Buffer

This register is the read buffer for the sub-counter RTCAnSUBC.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <RTCAn_base> + $10_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | RTCAnSRBU[21:16] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RTCAnSRBU[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 27.13 RTCAnSRBU Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 22 | Reserved | When read, the value after reset is returned. |
| 21 to 0 | RTCAnSRBU [21:0] | Sub-counter value at the time of the last RTCAnSUBC read.<br><br>When RTCAnCTL2.RTCAnRSUB is set to 1, the value of the RTCAnSUBC is loaded to the read buffer in synchronization with RTCATCKI. |

**NOTE**

Perform RTCAnSRBU read according to the flow described in **Section 27.5.4, Reading RTCAnSRBU**.

### 27.3.3.3 RTCAnSUBU — RTCA Clock Error Correction Register

This register enables and specifies clock error correction. This register only applies in 32.768-kHz mode (RTCAnCTL0.RTCAnSLSB = 0).

For details on clock error correction, see **Section 27.4.4, Clock Error Correction**.

**Access:** This register can be read or written in 8-bit units.
Note the following when writing this register during sub-counter operation:
- Previous RTCAnSUBU write must be completed (RTCAnCTL2.RTCAnWUST = 0).
- The write operation ends with the next sub-counter overflow.

**Address:** <RTCAn_base> + $38_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RTCAnDEV | RTCAnF6 | RTCAnF[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.14 RTCAnSUBU Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RTCAnDEV | Specifies how often clock error correction is performed per minute:<br>0: Three times every minute (when RTCAnSECC equals 00, 20, and 40)<br>1: Once every minute (when RTCAnSECC equals 00) |
| 6 | RTCAnF6 | Specifies whether the sub-counter value is incremented or decremented:<br>0: Incremented (+ correction)<br>Incrementation value = (RTCAnF[5:0] value – 1) × 2<br>1: Decremented (– correction)<br>Decrementation value = (inverted data of RTCAnF[5:0] value + 1) × 2 |
| 5 to 0 | RTCAnF[5:0] | Error correction value |

**NOTES**

1. When RTCAnF[5:1] = $00000_B$, clock error correction is not performed.
2. Perform RTCAnSUBU write as described in
   - **Section 27.5.1, Initial Setting of the RTCA**, and
   - **Section 27.5.5, Writing to RTCAnSUBU**.

### 27.3.3.4 RTCAnSCMP — RTCA Sub-Counter Compare Register

This register sets the compare value of the sub-counter RTCAnSUBC in frequency selection mode (RTCAnCTL0.RTCAnSLSB = 1).

When the sub-counter values matches the value of this register, an overflow signal is output to the seconds counter RTCAnSECC and the sub-counter is cleared.

Set the value for this register according to the frequency of the input clock RTCATCKI.

**Access:** This register can be read or written in 32-bit units.
Note the following when writing this register during sub-counter operation:
- Previous RTCAnSCMP write must be completed (RTCAnCTL2.RTCAnWSST = 0).
- The write operation ends with the next sub-counter overflow.

**Address:** <RTCAn_base> + $3C_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | RTCAnSCMP[21:16] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RTCAnSCMP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.15 RTCAnSCMP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 22 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 21 to 0 | RTCAnSCMP [21:0] | Sub-counter compare value in frequency selection mode. |

#### Example

The following example illustrates the setting of RTCAnSCMP:

- RTCATCKI = 4 MHz = 4,000,000 Hz
- RTCAnSCMP = 4,000,000 – 1 = 3,999,999 (decimal code) = $3D08FF_H$
- The seconds counter RTCAnSECC is triggered when the sub-counter value changes from $3D08FF_H$ to $0_H$.

**NOTES**

1. The operation of the RTCA cannot be guaranteed if a value of 3198 (decimal code) or lower is set in this register.
2. Perform RTCAnSCMP write as described in **Section 27.5.1, Initial Setting of the RTCA** and **Section 27.5.6, Writing to RTCAnSCMP**.

## 27.3.4 Details of RTCA Clock Counter and Buffer Registers

### 27.3.4.1 RTCAnSECC — RTCA Seconds Count Register

This register is the seconds counter. It counts seconds from 00 to 59 in BCD.

This register counts as follows.

- It is triggered by every overflow of the sub-counter RTCAnSUBC.

  If the sub-counter overflows while the seconds counter is stopped (RTCAnCTL2.RTCAnWST = 1), the seconds counter behaves as follows:

  – If one sub-counter overflow occurs while the seconds counter is stopped, the overflow is held internally.
  The seconds counter is incremented by one when it is restarted.

  – If two or more overflows occur while the seconds counter is stopped, the overflow count cannot be held internally.
  The seconds counter is incremented by one when it is restarted.

  – If the seconds counter was updated while the seconds counter is stopped, the sub-counter overflow(s) are ignored.

- It outputs an overflow signal when the value changes from 59 to 00. The overflow signal triggers the minutes counter (RTCAnMINC).

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <RTCAn_base> + $4C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | RTCAnSECC[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 27.16 RTCAnSECC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. |
| 6 to 0 | RTCAnSECC [6:0] | Seconds in BCD |

**NOTES**

1. Perform RTCAnSECC read according to the flow described in **Section 27.5.3, Reading Clock Counters**.
2. A start value can be assigned to this register by writing to the seconds count buffer register RTCAnSEC or to the clock time setting register RTCAnTIME. See
   - **Section 27.5.1, Initial Setting of the RTCA**, and
   - **Section 27.5.2, Updating Clock Counters**

### 27.3.4.2 RTCAnSEC — RTCA Seconds Count Buffer Register

This register is a buffer register to read/write the seconds counter RTCAnSECC.

**Access:** This register can be read or written in 8-bit units.

**Address:** <RTCAn_base> + $14_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | RTCAnSEC[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.17 RTCAnSEC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 6 to 0 | RTCAnSEC [6:0] | Seconds in BCD |

**NOTES**

1. When writing this register, only decimal values between 00 and 59 in BCD are allowed.
2. Perform RTCAnSEC read/write as described in
   - **Section 27.5.1, Initial Setting of the RTCA,**
   - **Section 27.5.2, Updating Clock Counters**, and
   - **Section 27.5.3, Reading Clock Counters.**

### 27.3.4.3 RTCAnMINC — RTCA Minutes Count Register

This register is the minutes counter. It counts minutes from 00 to 59 in BCD.

This register counts as follows.

- It is triggered by every overflow of the seconds counter RTCAnSECC.
- It outputs an overflow signal when the value changes from 59 to 00. The overflow signal triggers the hours counter (RTCAnHOURC).

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <RTCAn_base> + $50_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | RTCAnMINC[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 27.18 RTCAnMINC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. |
| 6 to 0 | RTCAnMINC [6:0] | Minutes in BCD |

**NOTES**

1. Perform RTCAnMINC read according to the flow described in **Section 27.5.3, Reading Clock Counters**.
2. A start value can be assigned to this register by writing to the minutes count buffer register RTCAnMIN or to the time count buffer register RTCAnTIME. See
   - **Section 27.5.1, Initial Setting of the RTCA**, and
   - **Section 27.5.2, Updating Clock Counters.**

### 27.3.4.4 RTCAnMIN — RTCA Minutes Count Buffer Register

This register is a buffer register to read/write the minutes counter RTCAnMINC.

**Access:** This register can be read or written in 8-bit units.

**Address:** <RTCAn_base> + $18_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | RTCAnMIN[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.19 RTCAnMIN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 6 to 0 | RTCAnMIN [6:0] | Minutes in BCD |

**NOTES**

1. When writing this register, only decimal values between 00 and 59 in BCD are allowed.
2. Perform RTCAnMIN read/write as described in
   - **Section 27.5.1, Initial Setting of the RTCA,**
   - **Section 27.5.2, Updating Clock Counters**, and
   - **Section 27.5.3, Reading Clock Counters.**

### 27.3.4.5 RTCAnHOURC — RTCA Hours Count Register

This register is the hours counter. It counts the hours in BCD. The count range depends on the selected hour format. See **Table 27.21, 12- and 24-Hour Format**.

This register counts as follows.

- It is triggered by every overflow of the minutes counter RTCAnMINC.
- It outputs an overflow signal when the value changes from 23 to 00 (in 24-hour format) or from 31 to 12 (in 12-hour format). The overflow signal triggers two counters:
  - Day of the week counter (RTCAnWEEKC)
  - Day of the month counter (RTCAnDAYC)

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <RTCAn_base> + $54_H$

**Value after reset:** $12_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | RTCAnHOURC[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 27.20 RTCAnHOURC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned. |
| 5 to 0 | RTCAnHOURC [5:0] | Hours in BCD.<br>See **Table 27.21, 12- and 24-Hour Format**, for details. |

**NOTES**

1. Perform RTCAnHOURC read according to the flow described in **Section 27.5.3, Reading Clock Counters**.
2. A start value can be assigned to this register by writing to the hours count buffer register RTCAnHOUR or to the time count buffer register RTCAnTIME. See
   - **Section 27.5.1, Initial Setting of the RTCA**, and
   - **Section 27.5.2, Updating Clock Counters.**

### 12- or 24-hour format

The count values of RTCAnHOURC depend on the selected hour format.

If 12-hour format is selected (RTCAnCTL0.RTCAnAMPM = 0), bit 5 in the RTCAnHOURC register is the am/pm indicator:

- RTCAnHOURC[5] = 0: am
- RTCAnHOURC[5] = 1: pm

The following table shows the count range of RTCAnHOURC in both 12- and 24-hour format.

**Table 27.21 12- and 24-Hour Format**

| 12-Hour Format (RTCAnAMPM = 0) | | | 24-Hour Format (RTCAnAMPM = 1) | |
|---|---|---|---|---|
| Time | RTCAnHOURC | | Time | RTCAnHOURC |
| 0 am | $12_H$ | | 0 | $00_H$ |
| 1 am | $01_H$ | | 1 | $01_H$ |
| 2 am | $02_H$ | | 2 | $02_H$ |
| 3 am | $03_H$ | | 3 | $03_H$ |
| 4 am | $04_H$ | | 4 | $04_H$ |
| 5 am | $05_H$ | | 5 | $05_H$ |
| 6 am | $06_H$ | | 6 | $06_H$ |
| 7 am | $07_H$ | | 7 | $07_H$ |
| 8 am | $08_H$ | | 8 | $08_H$ |
| 9 am | $09_H$ | | 9 | $09_H$ |
| 10 am | $10_H$ | | 10 | $10_H$ |
| 11 am | $11_H$ | | 11 | $11_H$ |
| 0 pm | $32_H$ | ↓ pm indicator in 12-hour format: RTCAnHOUR.RTCAnHOUR[5] = 1 | 12 | $12_H$ |
| 1 pm | $21_H$ | | 13 | $13_H$ |
| 2 pm | $22_H$ | | 14 | $14_H$ |
| 3 pm | $23_H$ | | 15 | $15_H$ |
| 4 pm | $24_H$ | | 16 | $16_H$ |
| 5 pm | $25_H$ | | 17 | $17_H$ |
| 6 pm | $26_H$ | | 18 | $18_H$ |
| 7 pm | $27_H$ | | 19 | $19_H$ |
| 8 pm | $28_H$ | | 20 | $20_H$ |
| 9 pm | $29_H$ | | 21 | $21_H$ |
| 10 pm | $30_H$ | | 22 | $22_H$ |
| 11 pm | $31_H$ | | 23 | $23_H$ |

### 27.3.4.6 RTCAnHOUR — RTCA Hours Count Buffer Register

This register is a buffer register to read/write the hours counter RTCAnHOURC.

**Access:** This register can be read or written in 8-bit units.

**Address:** <RTCAn_base> + $1C_H$

**Value after reset:** $12_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | RTCAnHOUR[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.22 RTCAnHOUR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 to 0 | RTCAnHOUR [5:0] | Hours in BCD<br>See **Table 27.21, 12- and 24-Hour Format**, for details. |

**NOTES**

1. When writing this register, only the following decimal values in BCD are allowed:
   - 12-hour format (RTCAnCTL0.RTCAnAMPM = 0):
     01 to 12 or 21 to 32
   - 24-hour format (RTCAnCTL0.RTCAnAMPM = 1):
     00 to 23
2. Perform RTCAnHOUR read/write as described in
   - **Section 27.5.1, Initial Setting of the RTCA,**
   - **Section 27.5.2, Updating Clock Counters**, and
   - **Section 27.5.3, Reading Clock Counters.**

### 27.3.4.7 RTCAnWEEKC — RTCA Day of the Week Count Register

This register is the day of the week counter. It counts from 0 to 6.

This register counts as follows.

- It is triggered by every overflow of the hours counter RTCAnHOURC.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <RTCAn_base> + $58_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | RTCAnWEEKC[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 27.23 RTCAnWEEKC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 3 | Reserved | When read, the value after reset is returned. |
| 2 to 0 | RTCAnWEEKC [2:0] | Day of the week |

**NOTES**

1. Perform RTCAnWEEKC read according to the flow described in **Section 27.5.3, Reading Clock Counters**.
2. A start value can be assigned to this register by writing to the day of the week count buffer register RTCAnWEEK or to the calendar count buffer register RTCAnCAL. See
   - **Section 27.5.1, Initial Setting of the RTCA**, and
   - **Section 27.5.2, Updating Clock Counters.**

### 27.3.4.8 RTCAnWEEK — RTCA Day of the Week Count Buffer Register

This register is a buffer register to read/write the day of the week counter RTCAnWEEKC.

There is no particular correspondence between the value of RTCAnWEEK and the day of the week. Set the correspondence according to the application to be used.

Example: 0 = Sunday, 1 = Monday, …, 6 = Saturday

**Access:** This register can be read or written in 8-bit units.

**Address:** <RTCAn_base> + $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | RTCAnWEEK[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W |

**Table 27.24 RTCAnWEEK Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 to 0 | RTCAnWEEK [2:0] | Day of the week |

**NOTES**

1. When writing this register, only decimal values between 0 and 6 in BCD are allowed.
2. Perform RTCAnWEEK read/write as described in
   - **Section 27.5.1, Initial Setting of the RTCA,**
   - **Section 27.5.2, Updating Clock Counters**, and
   - **Section 27.5.3, Reading Clock Counters.**

### 27.3.4.9 RTCAnDAYC — RTCA Day of the Month Count Register

This register is the day of the month counter. It counts from 01 to a maximum of 31 in BCD, depending on the value of the month counter (RTCAnMONC) and the year counter (RTCAnYEARC):

- 01 to 31 (January, March, May, July, August, October, December)
- 01 to 30 (April, June, September, November)
- 01 to 29 (February, leap year)
- 01 to 28 (February, non-leap year)

Years 0, 4, 8, 12, etc., are considered leap years.

This register counts as follows.

- It is triggered by every overflow of the hours counter RTCAnHOURC.
- It outputs an overflow signal when the value changes from 28, 29, 30, or 31 to 01, depending on the current month and year. The overflow signal triggers the month counter (RTCAnMONC).

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <RTCAn_base> + $5C_H$

**Value after reset:** $01_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | RTCAnDAYC[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R |

**Table 27.25 RTCAnDAYC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned. |
| 5 to 0 | RTCAnDAYC [5:0] | Day of the month in BCD |

**NOTES**

1. Perform RTCAnDAYC read according to the flow described in **Section 27.5.3, Reading Clock Counters**.
2. A start value can be assigned to this register by writing to the day of the month count buffer register RTCAnDAY or to the calendar count buffer register RTCAnCAL. See
   - **Section 27.5.1, Initial Setting of the RTCA**, and
   - **Section 27.5.2, Updating Clock Counters.**

### 27.3.4.10 RTCAnDAY — RTCA Day of the Month Count Buffer Register

This register is a buffer register to read/write the day of the month counter RTCAnDAYC.

**Access:** This register can be read or written in 8-bit units.

**Address:** <RTCAn_base> + $24_H$

**Value after reset:** $01_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | RTCAnDAY[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.26 RTCAnDAY Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 to 0 | RTCAnDAY [5:0] | Day of the month in BCD |

**NOTES**

1. When writing this register, only decimal values between 01 and 31 in BCD are allowed:
   - 01 to 31 (January, March, May, July, August, October, December)
   - 01 to 30 (April, June, September, November)
   - 01 to 29 (February, leap year)
   - 01 to 28 (February, non-leap year)
2. Perform RTCAnDAY read/write as described in
   - **Section 27.5.1, Initial Setting of the RTCA,**
   - **Section 27.5.2, Updating Clock Counters**, and
   - **Section 27.5.3, Reading Clock Counters.**

### 27.3.4.11 RTCAnMONC — RTCA Month Count Register

This register is the month counter. It counts the month of the year, starting from 01 to 12 in BCD.

This register counts as follows.

- It is triggered by every overflow of the counter for the day of the month RTCAnDAYC.
- It outputs an overflow signal when the value changes from 12 to 01. The overflow signal triggers the year counter (RTCAnYEARC).

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <RTCAn_base> + $60_H$

**Value after reset:** $01_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | RTCAnMONC[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R |

**Table 27.27 RTCAnMONC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 5 | Reserved | When read, the value after reset is returned. |
| 4 to 0 | RTCAnMONC [4:0] | Month of the year in BCD |

**NOTES**

1. Perform RTCAnMONC read according to the flow described in **Section 27.5.3, Reading Clock Counters**.
2. A start value can be assigned to this register by writing to the month count buffer register RTCAnMONTH or to the calendar count buffer register RTCAnCAL. See
   - **Section 27.5.1, Initial Setting of the RTCA**, and
   - **Section 27.5.2, Updating Clock Counters.**

### 27.3.4.12 RTCAnMONTH — RTCA Month Count Buffer Register

This register is a buffer register to read/write the month counter RTCAnMONC.

**Access:** This register can be read or written in 8-bit units.

**Address:** <RTCAn_base> + $28_H$

**Value after reset:** $01_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | RTCAnMONTH[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 27.28 RTCAnMONTH Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 to 0 | RTCAnMONTH [4:0] | Month of the year in BCD |

**NOTES**

1. When writing this register, only decimal values between 01 and 12 in BCD are allowed.
2. Perform RTCAnMONTH read/write as described in
   - **Section 27.5.1, Initial Setting of the RTCA,**
   - **Section 27.5.2, Updating Clock Counters**, and
   - **Section 27.5.3, Reading Clock Counters.**

### 27.3.4.13 RTCAnYEARC — RTCA Year Count Register

This register is the year counter. It counts years from 00 to a maximum of 99 in BCD.

Years 00, 04, 08, …, 92, and 96 (every four years) are considered leap years.

This register counts as follows.

- It is triggered by every overflow of the month counter RTCAnMONC.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <RTCAn_base> + $64_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RTCAnYEARC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 27.29 RTCAnYEARC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | RTCAnYEARC [7:0] | Year in BCD |

**NOTES**

1. Perform RTCAnYEARC read according to the flow described in **Section 27.5.3, Reading Clock Counters**.
2. A start value can be assigned to this register by writing to the year count buffer register RTCAnYEAR or to the calendar count buffer register RTCAnCAL. See
   - **Section 27.5.1, Initial Setting of the RTCA**, and
   - **Section 27.5.2, Updating Clock Counters.**

### 27.3.4.14 RTCAnYEAR — RTCA Year Count Buffer Register

This register is a buffer register to read/write the year counter RTCAnYEARC.

**Access:** This register can be read or written in 8-bit units.

**Address:** <RTCAn_base> + $2C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RTCAnYEAR[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.30 RTCAnYEAR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | RTCAnYEAR [7:0] | Year in BCD |

**NOTES**

1. When writing this register, only decimal values between 00 and 99 in BCD are allowed.
2. Perform RTCAnYEAR read/write as described in
   - **Section 27.5.1, Initial Setting of the RTCA,**
   - **Section 27.5.2, Updating Clock Counters**, and
   - **Section 27.5.3, Reading Clock Counters.**

## 27.3.5 Details of RTCA Special Counter and Buffer Registers

### 27.3.5.1 RTCAnTIMEC — RTCA Time Count Register

This register enables the RTCAnHOURC, RTCAnMINC, and RTCAnSECC counters to be read simultaneously.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <RTCAn_base> + $68_H$

**Value after reset:** $0012\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | RTCAnHOURC[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | RTCAnMINC[6:0] | | | | | | | — | RTCAnSECC[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 27.31 RTCAnTIMEC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 22 | Reserved | When read, the value after reset is returned. |
| 21 to 16 | RTCAnHOURC [5:0] | Hours in BCD.<br>See **Table 27.21, 12- and 24-Hour Format**, for details. |
| 15 | Reserved | When read, the value after reset is returned. |
| 14 to 8 | RTCAnMINC [6:0] | Minutes in BCD |
| 7 | Reserved | When read, the value after reset is returned. |
| 6 to 0 | RTCAnSECC [6:0] | Seconds in BCD |

**NOTES**

1. Perform RTCAnTIMEC read according to the flow described in **Section 27.5.3, Reading Clock Counters**.
2. A start value can be assigned to this register by writing to the time count buffer register RTCAnTIME. See
   - **Section 27.5.1, Initial Setting of the RTCA**, and
   - **Section 27.5.2, Updating Clock Counters.**

### 27.3.5.2 RTCAnTIME — RTCA Time Count Buffer Register

This register enables the RTCAnHOUR, RTCAnMIN, and RTCAnSEC buffer registers to be read/written simultaneously.

**Access:** This register can be read or written in 32-bit units.

**Address:** <RTCAn_base> + $30_H$

**Value after reset:** $0012\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | RTCAnHOUR[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | RTCAnMIN[6:0] | | | | | | | — | RTCAnSEC[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.32 RTCAnTIME Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 22 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 21 to 16 | RTCAnHOUR [5:0] | Hours in BCD<br>See **Table 27.21, 12- and 24-Hour Format**, for details. |
| 15 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 14 to 8 | RTCAnMIN [6:0] | Minutes in BCD |
| 7 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 6 to 0 | RTCAnSEC [6:0] | Seconds in BCD |

**NOTE**

Perform RTCAnTIME read/write as described in

- **Section 27.5.1, Initial Setting of the RTCA,**
- **Section 27.5.2, Updating Clock Counters**, and
- **Section 27.5.3, Reading Clock Counters.**

### 27.3.5.3 RTCAnCALC — RTCA Calendar Count Register

This register enables the RTCAnYEARC, RTCAnMONC, RTCAnDAYC, and RTCAnWEEKC counters to be read simultaneously.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <RTCAn_base> + $6C_H$

**Value after reset:** $0001\ 0100_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RTCAnYEARC[7:0] | | | | | | | | — | — | — | RTCAnMONC[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | RTCAnDAYC[5:0] | | | | | | — | — | — | — | — | RTCAnWEEKC[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 27.33 RTCAnCALC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RTCAnYEARC [7:0] | Year in BCD |
| 23 to 21 | Reserved | When read, the value after reset is returned. |
| 20 to 16 | RTCAnMONC [4:0] | Month of the year in BCD |
| 15, 14 | Reserved | When read, the value after reset is returned. |
| 13 to 8 | RTCAnDAYC [5:0] | Day of the month in BCD |
| 7 to 3 | Reserved | When read, the value after reset is returned. |
| 2 to 0 | RTCAnWEEKC [2:0] | Day of the week in BCD |

**NOTES**

1. Perform RTCAnCALC read according to the flow described in **Section 27.5.3, Reading Clock Counters**.
2. A start value can be assigned to this register by writing to the clock time setting register RTCAnCAL. See
   - **Section 27.5.1, Initial Setting of the RTCA**, and
   - **Section 27.5.2, Updating Clock Counters.**

### 27.3.5.4 RTCAnCAL — RTCA Calendar Count Buffer Register

This register enables the RTCAnYEAR, RTCAnMONTH, RTCAnDAY, and RTCAnWEEK buffer registers to be read/written simultaneously.

**Access:** This register can be read or written in 32-bit units.

**Address:** <RTCAn_base> + $34_H$

**Value after reset:** $0001\ 0100_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RTCAnYEAR[7:0] | | | | | | | | — | — | — | RTCAnMONTH[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | RTCAnDAY[5:0] | | | | | | — | — | — | — | — | RTCAnWEEK[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

**Table 27.34 RTCAnCAL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | RTCAnYEAR [7:0] | Year in BCD |
| 23 to 21 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 20 to 16 | RTCAnMONTH [4:0] | Month of the year in BCD |
| 15, 14 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 13 to 8 | RTCAnDAY [5:0] | Day of the month in BCD |
| 7 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 to 0 | RTCAnWEEK [2:0] | Day of the week in BCD |

**NOTE**

Perform RTCAnCAL read/write as described in

- **Section 27.5.1, Initial Setting of the RTCA.**
- **Section 27.5.2, Updating Clock Counters**, and
- **Section 27.5.3, Reading Clock Counters.**

## 27.3.6 Details of RTCA Alarm Setting Registers

### 27.3.6.1 RTCAnALM — RTCA Alarm Minute Setting Register

This register specifies the minute of the alarm interrupt.

For details and example settings, see **Section 27.4.3, Alarm Interrupt Function**.

**Access:** This register can be read or written in 8-bit units.

**Address:** <RTCAn_base> + $40_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | RTCAnALM[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.35 RTCAnALM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 6 to 0 | RTCAnALM [6:0] | Minute of the alarm interrupt in BCD |

**NOTES**

1. If decimal values outside the range of 00 to 59 in BCD are set, no alarm interrupt request will be generated.
2. When the setting of RTCAnALM is changed during sub-counter operation (RTCAnCTL0.RTCAnCEST = 1), a glitch may be output to RTCATINTAL. Implement appropriate interrupt mask processing procedures.

### 27.3.6.2 RTCAnALH — RTCA Alarm Hour Setting Register

This register specifies the hour of the alarm interrupt.

For details and example settings, see **Section 27.4.3, Alarm Interrupt Function**.

**Access:** This register can be read or written in 8-bit units.

**Address:** <RTCAn_base> + $44_H$

**Value after reset:** $12_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | RTCAnALH[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.36 RTCAnALH Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 to 0 | RTCAnALH [5:0] | Hour of the alarm interrupt in BCD |

**NOTES**

1. If decimal values outside the following range are set, no alarm interrupt request will be generated:
   - 12-hour format (RTCAnCTL0.RTCAnAMPM = 0): 01 to 12 or 21 to 32
   - 24-hour format (RTCAnCTL0.RTCAnAMPM = 1): 00 to 23
2. When the setting of RTCAnALH is changed during sub-counter operation (RTCAnCTL0.RTCAnCEST = 1), a glitch may be output to RTCATINTAL. Implement appropriate interrupt mask processing procedures.

### 27.3.6.3 RTCAnALW — RTCA Alarm Day of the Week Setting Register

This register specifies the day(s) of the week of the alarm interrupt.

**Access:** This register can be read or written in 8-bit units.

**Address:** <RTCAn_base> + $48_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | RTCAnALW[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.37 RTCAnALW Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 6 to 0 | RTCAnALW [6:0] | Specifies day of the week m (m = 0 to 6) as a day, when an alarm interrupt request is generated:<br>0: No alarm interrupt request is generated on day m.<br>1: Alarm interrupt request is generated on day m at the time set using RTCAnALM and RTCAnALH.<br>The bits of this register correspond to the count value of the day of the week counter (RTCAnWEEKC). |

**NOTE**

When the setting of RTCAnALW is changed during sub-counter operation (RTCAnCTL0.RTCAnCE = 1), a glitch may be output to RTCATINTAL. Implement appropriate interrupt mask processing procedures.

#### Example

If Sunday is RTCAnWEEK = 0, Monday is RTCAnWEEK = 1, Tuesday is RTCAnWEEK = 2, …, Saturday is RTCAnWEEK = 6:

- To set the alarm for Sunday, set RTCAnALW = $0000\ 0001_B$.
- To set the alarm for Monday and Wednesday, set RTCAnALW = $0000\ 1010_B$.
- To set the alarm for Tuesday, Thursday, and Saturday, set RTCAnALW = $0101\ 0100_B$.

For more examples, see **Section 27.4.3, Alarm Interrupt Function**.

## 27.3.7 RTCA Emulation Register

### 27.3.7.1 RTCAnEMU — RTCA Emulation Register

This register controls operation by SVSTOP.

**Access:** This register can be read or written in 8-bit or 1-bit units.
A write should be performed when EPC.SVSTOP = 0.

**Address:** <RTCAn_base> + $74_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RTCAnSVSDIS | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R |

**Table 27.38 RTCAnEMU Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RTCAnSVSDIS | When the EPC.SVSTOP bit is set to 0:<br>Count clock is supplied when the debugger gains microcontroller control (at a breakpoint, etc.) regardless of the value of this bit.<br><br>When the EPC.SVSTOP bit is set to 1:<br>0: Count clock is stopped when the debugger gains microcontroller control (as at a breakpoint).<br>1: Count clock continues to be supplied when the debugger gains microcontroller control (at a breakpoint, etc.). |
| 6 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

## 27.4 Operation

The RTCA provides two operation modes:

- Frequency selection mode
- 32.768-kHz mode

The operation mode that can be used depends on the available input clock RTCATCKI. The operation mode specifies the sub-counter compare value that is used to trigger the seconds counter and thus all subsequent counters. Clock error correction is only possible in 32.768-kHz mode.

The following table provides an overview of the properties of the two operation modes.

**Table 27.39 RTCA Operation Mode Overview**

| | | 32.768-kHz Mode | |
|---|---|---|---|
| | **Frequency Selection Mode** | **Clock Correction Disabled** | **Clock Correction Enabled** |
| Allowed input clock RTCATCKI | Any frequency from 32 kHz to 4.194304 MHz | 32.768 kHz | Any frequency from 32.76180000 kHz to 32.77420000 kHz |
| Sub-counter RTCAnSUBC operation | • Counter overflow at value of RTCAnSCMP<br>• RTCAnSCMP must be set to RTCATCKI-1 (in Hz) | Counter overflow at $7FFF_H$ | Counter overflow at $7FFF_H$ or<br>Every 20 or 60 seconds: $7FFF_H$ ±RTCAnSUBU.RTCAnF[5:0] |

The operation mode is selected by control bit RTCAnCTL0.RTCAnSLSB. For details on how to set the operation mode during RTCA initialization, see **Section 27.5.1, Initial Setting of the RTCA**.

**CAUTIONS**

1. **The input clock RTCATCKI must not be outside the allowed frequency range.**
2. **The operation mode must not be changed while sub-counter operation is enabled (RTCAnCTL0.RTCAnCEST = 1).**

### 27.4.1 Clock Counter Format

The clock counters (RTCAnSECC to RTCAnYEARC) operate on binary coded decimals (BCD): Each digit is represented by its own binary sequence.

Depending on the valid data range, the number of bits for a digit differs. For example, the tens digit of the month of the year counter has only one bit (for 0 and 1) whereas the tens digit of the minutes counter has 3 bits (for 0 to 5).

The following table lists the decimals 0 to 59 in binary and BCD.

**Table 27.40 Example of BCD Code – Seconds or Minutes Counter (0 to 59)**

| Decimal | Binary | BCD |
|---|---|---|
| 0 | 000000 | 000 0000 |
| 1 | 000001 | 000 0001 |
| 2 | 000010 | 000 0010 |
| 3 | 000011 | 000 0011 |
| 4 | 000100 | 000 0100 |
| 5 | 000101 | 000 0101 |
| 6 | 000110 | 000 0110 |
| 7 | 000111 | 000 0111 |
| 8 | 001000 | 000 1000 |
| 9 | 001001 | 000 1001 |
| 10 | 001010 | 001 0000 |
| 11 | 001011 | 001 0001 |
| 12 | 001100 | 001 0010 |
| : | : | : |
| 58 | 111010 | 101 1000 |
| 59 | 111011 | 101 1001 |

### 27.4.2 Fixed Interval Interrupt Function

Interrupt RTCATINTR can be specified to occur after every 0.25 seconds, 0.5 seconds, 1 (full) second, 1 (full) minute, 1 (full) hour, 1 (full) day, or 1 (full) month.

The fixed interval interrupt function is controlled by bits RTCAnCTL1.RTCAnCT[2:0].

### 27.4.3 Alarm Interrupt Function

Interrupt RTCATINTAL can be specified to occur at a certain time on one or several days of the week. This interrupt can be used as a wake-up signal.

The alarm interrupt function is enabled and disabled by bit RTCAnCTL1.RTCAnENALM.

The alarm setting is specified by the following control registers:

- RTCAnALW selects the weekday(s).

The allocation of bits to weekdays is defined by the day of the week count buffer register RTCAnWEEK.

- RTCAnALH and RTCAnALM specify the hour and minute in BCD.

### Examples

The following tables show some exemplary settings of the alarm control registers for both 12-hour and 24-hour format.

In this example, Sunday is RTCAnWEEK = 0, Monday is RTCAnWEEK = 1, Tuesday is RTCAnWEEK = 2, …, Saturday is RTCAnWEEK = 6:

**Table 27.41 Alarm Setting in 12-Hour Format (RTCAnCTL0.RTCAnAMPM = 0)**

| Alarm Setting Time | RTCAnALW | RTCAnALH | RTCAnALM |
|---|---|---|---|
| Sunday 7:00 am | $01_H$ | $07_H$ | $00_H$ |
| Sunday, Monday 12:15 pm | $03_H$ | $32_H$ | $15_H$ |
| Monday, Wednesday, Friday 5:30 pm | $2A_H$ | $25_H$ | $30_H$ |
| Daily, 10:45 pm | $7F_H$ | $30_H$ | $45_H$ |

**Table 27.42 Alarm Setting in 24-Hour Format (RTCAnCTL0.RTCAnAMPM = 1)**

| Alarm Setting Time | RTCAnALW | RTCAnALH | RTCAnALM |
|---|---|---|---|
| Sunday 7:00 | $01_H$ | $07_H$ | $00_H$ |
| Sunday, Monday 12:15 | $03_H$ | $12_H$ | $15_H$ |
| Monday, Wednesday, Friday 17:30 | $2A_H$ | $17_H$ | $30_H$ |
| Daily, 22:45 | $7F_H$ | $22_H$ | $45_H$ |

## 27.4.4 Clock Error Correction

Clock error correction compensates for deviations of the oscillator from the nominal clock rate. With clock error correction input clock rates from 32.76180 kHz to 32.77420 kHz are possible.

The clock error correction function is only available in 32.768-kHz operation mode. In this operation mode, a nominal clock rate of 32.768 kHz is expected and the sub-counters overflow value is fixed to $7FFF_H$.

The following figures illustrate the clock error when the input clock rate deviates from the nominal clock.

### RTCATCKI = 32.768 kHz

**Figure 27.2, RTCATCKI = 32.768 kHz, No Clock Error Correction Required** shows the timing diagram if RTCATCKI matches the nominal clock rate of 32.768 kHz. No clock error correction is required.

Counting from 0 to 32767 (0 to $7FFF_H$) with a 32.768-kHz clock is exactly equal to one second.

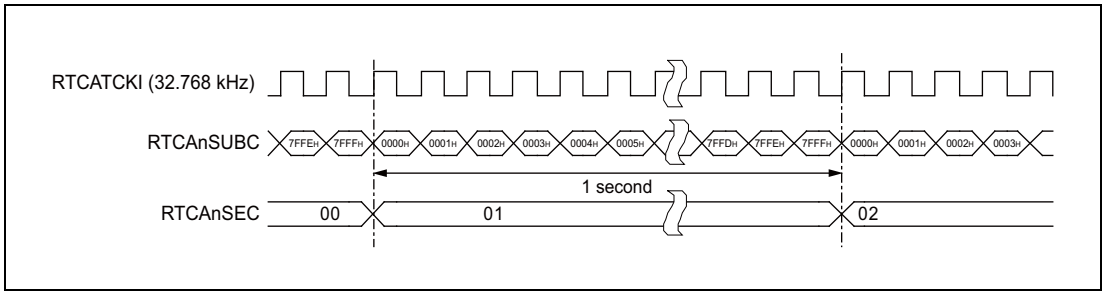



**Figure 27.2 RTCATCKI = 32.768 kHz, No Clock Error Correction Required**

### RTCATCKI = 32.769 kHz

**Figure 27.3, RTCATCKI = 32.769 kHz, No Clock Error Correction Enabled** shows the timing diagram if RTCATCKI deviates from the nominal clock rate of 32.768 kHz. In this example, RTCATCKI is connected to a 32.769-kHz oscillator. Clock error correction is not enabled.

Counting from 0 to 32767 (0 to $7FFF_H$) with a 32.769-kHz clock is equal to approximately 0.99997 seconds (32768/32769). A "+ error" (faster than 32.768-kHz) occurs. In one month, RTCA deviates approximately –79 seconds from the real time.

Error = (32768/32769 - 1) × 60 (s) × 60 (min) × 24 (h) × 30 (d)

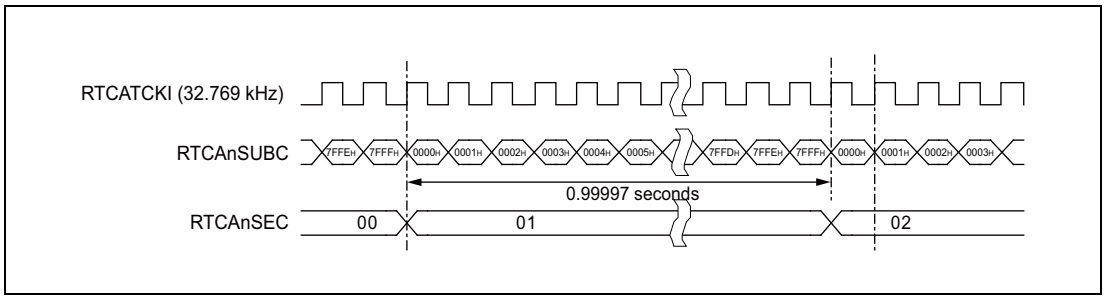



**Figure 27.3 RTCATCKI = 32.769 kHz, No Clock Error Correction Enabled**

Clock error correction is performed by stretching/reducing the 1-second period of the sub-counter at regular intervals. The sub-counter's upper limit of $7FFF_H$ is increased or decreased by setting the following parameters in register RTCAnSUBU:

- A correction value greater than one
- An operator (add/subtract)
- An interval (20 or 60 seconds)

The corrected overflow value becomes effective every 20 or 60 seconds, so that on the average RTCAnSECC is triggered exactly every second.

#### 27.4.4.1 Setting the Correction Value and the Operator

The correction value and operator are specified by the RTCAnF6, RTCAnF[5:0] bits of the RTCAnSUBU register:

- RTCAnF6 specifies whether the overflow value is incremented or decremented.
- RTCAnF[5:0] specifies the correction value.

The correction values are calculated as follows:

**Table 27.43 Correction Value Settings**

| RTCAnF6 | Increment/Decrement | Correction Value |
|---|---|---|
| 0 | Increment | (Value of RTCAnF[5:0] – 1) × 2 |
| 1 | Decrement | (Inverted value of RTCAnF[5:0] + 1) × 2 |

Some examples are given in the following table:

**Table 27.44 Correction Value Examples**

| RTCAnF6 | RTCAnF[5:0] | Correction Value | Count Limit of RTCAnSUBC |
|---|---|---|---|
| 0 | $15_H$ | $(15_H - 1) \times 2 = 40$ | 32768 + 40 = 32808 |
| 1 | $15_H$ | $(\overline{15}_H + 1) \times 2$<br>$= (2A_H + 1) \times 2$<br>$= 86$ | 32768 – 86 = 32682 |

#### 27.4.4.2 Impact of the Repetition Interval

The correction value set by RTCAnF6, RTCAnF[5:0] does not change the count limit of RTCAnSUBC every second. The repetition interval at which the correction value becomes effective is specified by bit RTCAnDEV.

This bit also influences the size of the correctable frequency range and the correction accuracy.

The following table summarizes the RTCAnDEV settings.

**Table 27.45 Setting of Bit RTCAnSUBU.RTCAnDEV**

| RTCAnDEV | Count Limit of RTCAnSUBC is Changed | Frequency Range that can be Corrected | Correction Accuracy |
|---|---|---|---|
| 0 | Every 20 seconds<br>when RTCAnSECC = 00, 20, or 40 | 32.76180000 to 32.77420000 kHz | |
| 1 | Every 60 seconds<br>when RTCAnSECC = 00 | 32.76593333 to 32.77006667 kHz | Three times higher than for RTCAnDEV = 0 |

### 27.4.4.3 Sample Settings

The frequencies that can be corrected, as well as the setting values of bits RTCAnDEV, RTCAnF6, and RTCAnF[5:0], are listed in the following table.

**Table 27.46 Correctable Frequency Range when RTCAnDEV = 0**

| Input Clock Frequency | RTCAnF6 | RTCAnF[5:0] | Correction Value of RTCAnSUBC |
|---|---|---|---|
| — | 0 | 000000 | No correction |
| — | 0 | 000001 | No correction |
| 32.76810000 kHz | 0 | 000010 | Once every 20 s,<br>RTCAnSUBC count value + 2 |
| 32.76820000 kHz | 0 | 000011 | Once every 20 s,<br>RTCAnSUBC count value + 4 |
| 32.76830000 kHz | 0 | 000100 | Once every 20 s,<br>RTCAnSUBC count value + 6 |
| : | : | : | : |
| 32.77400000 kHz | 0 | 111101 | Once every 20 s,<br>RTCAnSUBC count value + 120 |
| 32.77410000 kHz | 0 | 111110 | Once every 20 s,<br>RTCAnSUBC count value + 122 |
| 32.77420000 kHz<br>(upper limit) | 0 | 111111 | Once every 20 s,<br>RTCAnSUBC count value + 124 |
| — | 1 | 000000 | No correction |
| — | 1 | 000001 | No correction |
| 32.76180000 kHz<br>(lower limit) | 1 | 000010 | Once every 20 s,<br>RTCAnSUBC count value – 124 |
| 32.76190000 kHz | 1 | 000011 | Once every 20 s,<br>RTCAnSUBC count value – 122 |
| 32.76200000 kHz | 1 | 000100 | Once every 20 s,<br>RTCAnSUBC count value – 120 |
| : | : | : | : |
| 32.76770000 kHz | 1 | 111101 | Once every 20 s,<br>RTCAnSUBC count value – 6 |
| 32.76780000 kHz | 1 | 111110 | Once every 20 s,<br>RTCAnSUBC count value – 4 |
| 32.76790000 kHz | 1 | 111111 | Once every 20 s,<br>RTCAnSUBC count value – 2 |

**Table 27.47 Correctable Frequency Range when RTCAnDEV = 1**

| Input Clock Frequency | RTCAnF6 | RTCAnF[5:0] | Correction Value of RTCAnSUBC |
|---|---|---|---|
| — | 0 | 000000 | No correction |
| — | 0 | 000001 | No correction |
| 32.76803333 kHz | 0 | 000010 | Once every 60 s,<br>RTCAnSUBC count value + 2 |
| 32.76806667 kHz | 0 | 000011 | Once every 60 s,<br>RTCAnSUBC count value + 4 |
| 32.76810000 kHz | 0 | 000100 | Once every 60 s,<br>RTCAnSUBC count value + 6 |
| : | : | : | : |
| 32.77000000 kHz | 0 | 111101 | Once every 60 s,<br>RTCAnSUBC count value + 120 |
| 32.77003333 kHz | 0 | 111110 | Once every 60 s,<br>RTCAnSUBC count value + 122 |
| 32.77006667 kHz<br>(upper limit) | 0 | 111111 | Once every 60 s,<br>RTCAnSUBC count value + 124 |
| — | 1 | 000000 | No correction |
| — | 1 | 000001 | No correction |
| 32.76593333 kHz<br>(lower limit) | 1 | 000010 | Once every 60 s,<br>RTCAnSUBC count value – 124 |
| 32.76596667 kHz | 1 | 000011 | Once every 60 s,<br>RTCAnSUBC count value – 122 |
| 32.76600000 kHz | 1 | 000100 | Once every 60 s,<br>RTCAnSUBC count value – 120 |
| : | : | : | : |
| 32.76790000 kHz | 1 | 111101 | Once every 60 s,<br>RTCAnSUBC count value – 6 |
| 32.76793333 kHz | 1 | 111110 | Once every 60 s,<br>RTCAnSUBC count value – 4 |
| 32.76796667 kHz | 1 | 111111 | Once every 60 s,<br>RTCAnSUBC count value – 2 |

## 27.5 Procedures for Setup, Writing and Reading

The following subsections provide flow charts that illustrate the procedures for RTCA setup and for reading and writing the RTCA clock counters.

### 27.5.1 Initial Setting of the RTCA

The RTCA must be stopped before setting the initial setting value of each counter.

#### 27.5.1.1 RTCA Stop Procedure

Stop the RTCA according to the following flow.

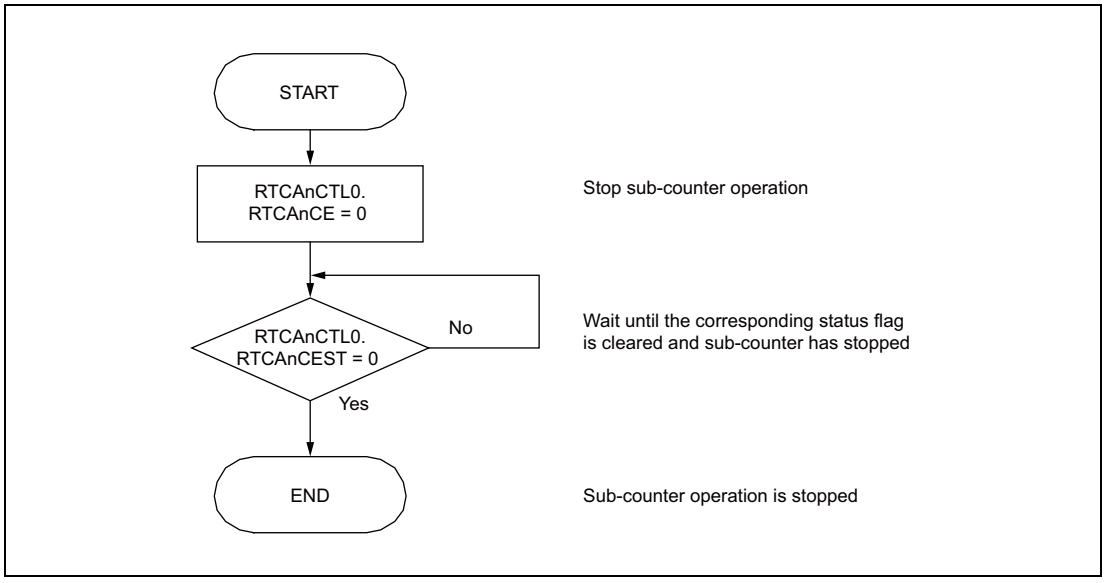



**Figure 27.4 RTCA Stop Procedure**

### 27.5.1.2 RTCA Initialization Procedure

Perform the initial setting of the RTCA according to the following flow:

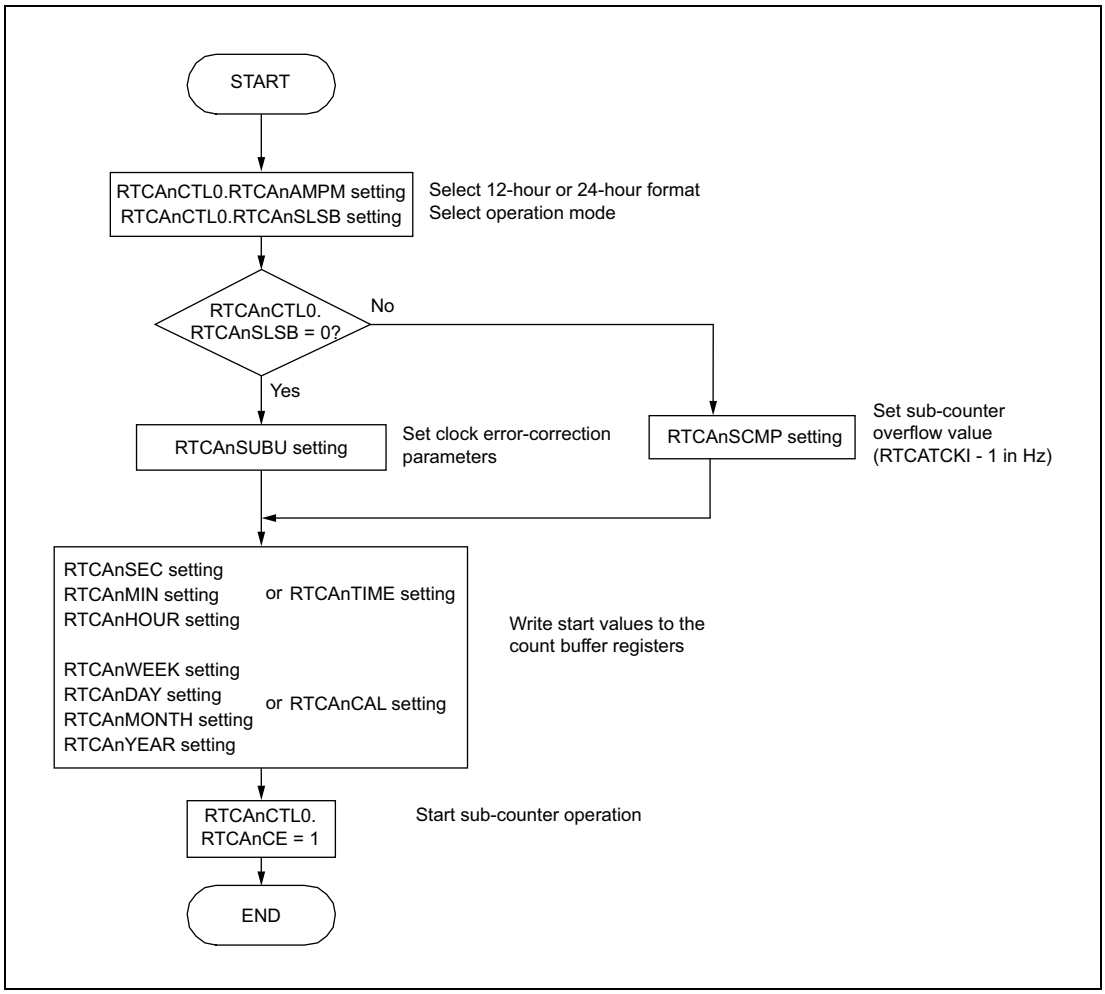



**Figure 27.5 RTCA Initial Setup Procedure**

**CAUTION**

**The internal clock counter is synchronized with RTCATCKI.**

**In addition, two RTCATCKI periods are required before the clock counter starting behind END of the above flow.**

**Therefore, PCLK must be continuously supplied until the completion of the initial setting.**

**Check that RTCAnCTL0.RTCAnCEST = 1, when the supply of PCLK is stopped after setting the initial setting value of RTCA.**

### 27.5.2 Updating Clock Counters

The clock counters RTCAnSECC to RTCAnYEARC can be stopped and updated while the sub-counter is running.

To update the clock counter when the sub-counter operation is enabled (RTCAnCTL0.RTCAnCE = 1), follow the flowchart shown below.

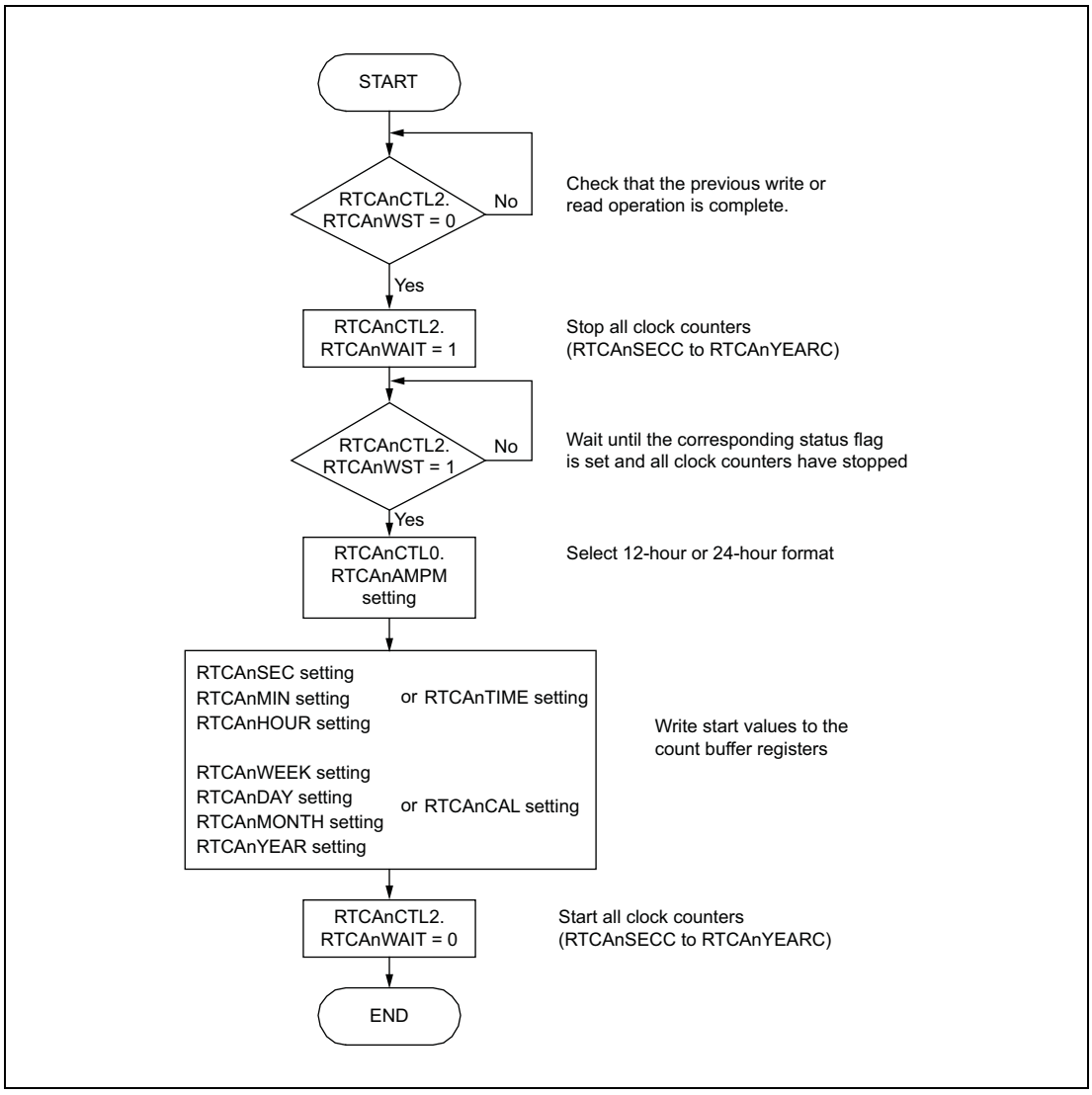



**Figure 27.6 Updating Clock Counter Values**

**CAUTIONS**

1. **The internal clock counter is synchronized with RTCATCKI.**
   **In addition, two RTCATCKI periods are required before the clock counter updating behind END of the above flow.**
   **Therefore, PCLK must be continuously supplied until the completion of the clock counter updating.**
   **Check that RTCAnCTL2.RTCAnWST = 0 before stopping the supply of PCLK after the completion of clock counter updating.**
2. **The update procedure must be completed within one second. Otherwise the Real-Time Clock will not count correctly any more:**

3. **Only one sub-counter overflow can be held internally and increment the seconds counter after restarting the clock counters if the value is held.**
4. **If the sub-counter overflows more than once during clock counter stop, the overflow count cannot be held internally. Thus the seconds counter is incremented by one instead of by two when it is restarted.**

### 27.5.3 Reading Clock Counters

There are two methods to read the clock counters while sub-counter operation is enabled:

- Reading count buffer registers
- Reading counter registers

The advantages and disadvantages of the two methods are summarized in the following table.

**Table 27.48 Comparison of the Two Read Methods**

| | Advantage | Disadvantage |
|---|---|---|
| Reading count buffer registers | It is unnecessary to read clock counters several times because the clock counters are read synchronously. | A program wait state occurs between setting RTCAnCTL2.RTCAnWAIT = 1 and completion of data transfer. |
| Reading count registers | Program wait state does not occur. | If the sub-counter increments, the clock counters must be read several times because they are read asynchronously to RTCATCKI. |

#### 27.5.3.1 Procedure for Reading Count Buffer Registers

The following operations are necessary:

1. Stop all clock counters (RTCAnCTL2.RTCAnWAIT = 1). The value of the clock counters is transferred to the corresponding count buffer registers.
2. Read the count buffer registers.

A program wait state occurs between setting RTCAnCTL2.RTCAnWAIT = 1 and completion of data transfer.

The maximum delay is three PCLK periods plus two RTCATCKI periods. For example, if the RTCA operates with PCLK = 40 MHz and RTCATCKI = 32.768 kHz, the delay is about 61 µs.

To read the count buffer register when the sub-counter operation is enabled (RTCAnCTL0.RTCAnCEST = 1), follow the flowchart shown below.

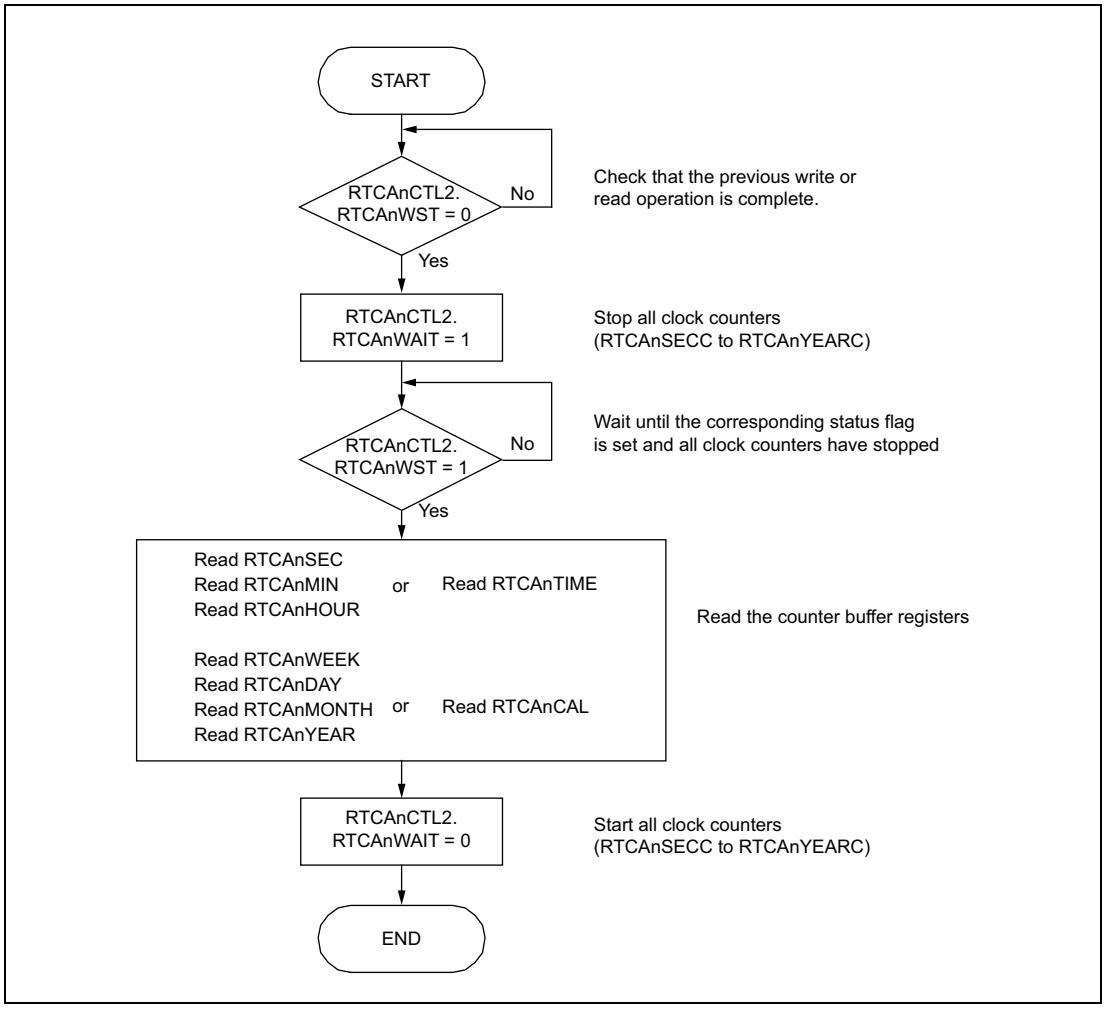



**Figure 27.7 Reading Clock Count Buffer Registers**

**CAUTIONS**

1. **The internal clock counter is synchronized with RTCATCKI.**
   **In addition, two RTCATCKI periods are required before resuming counter behind END of the above flow.**
   **Therefore, PCLK must be continuously supplied until the counter resuming.**
   **Check that RTCAnCTL0.RTCAnCEST = 1 first to stop the supply of PCLK after count buffer register reading.**
2. **The reading procedure must be completed within one second. Otherwise the Real-Time Clock will not count correctly any more.**
3. **Only one sub-counter overflow can be held internally. If there is a value held internally when the clock counter restarts, the seconds counter will be incremented by 1.**
4. **If the sub-counter overflows more than once during clock counter stop, the overflow count cannot be held internally. Thus the seconds counter is incremented by one instead of by two when it is restarted.**

### 27.5.3.2 Procedure for Reading Counter Registers Directly

To ensure that the sub-counter did not overflow while reading the counters, the seconds counter RTCAnSECC must be read twice in the beginning and at the end of the procedure. The first read value is compared with the second read value.

- First read value = second read value:
  No overflow of sub-counter occurred during counter read operation.
- First read value ≠ second read value:
  Overflow of the sub-counter occurred during counter read operation. The counters must be read again to get the current counter values.

To read the counter register directly when the sub-counter operation is enabled (RTCAnCTL0.RTCAnCE = 1), follow the flowchart shown below.

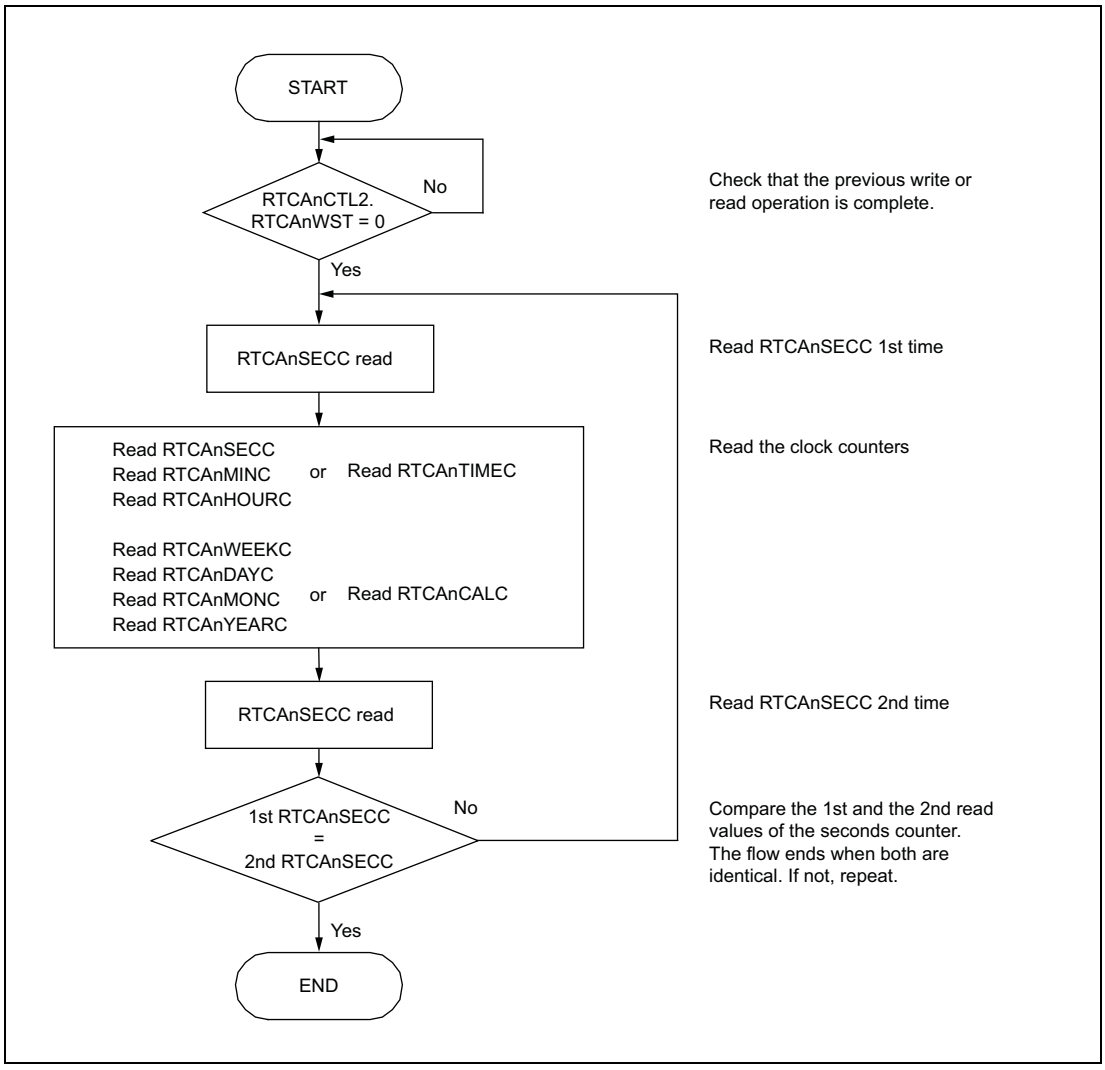



**Figure 27.8 Reading Clock Counter Registers**

**NOTE**

The procedure must be completed within one second.

### 27.5.4 Reading RTCAnSRBU

RTCAnSRBU is the read buffer register for the sub-counter.

When the sub-counter operation is enabled (RTCAnCTL0.RTCAnCE = 1), read RTCAnSRBU according to the following flow.

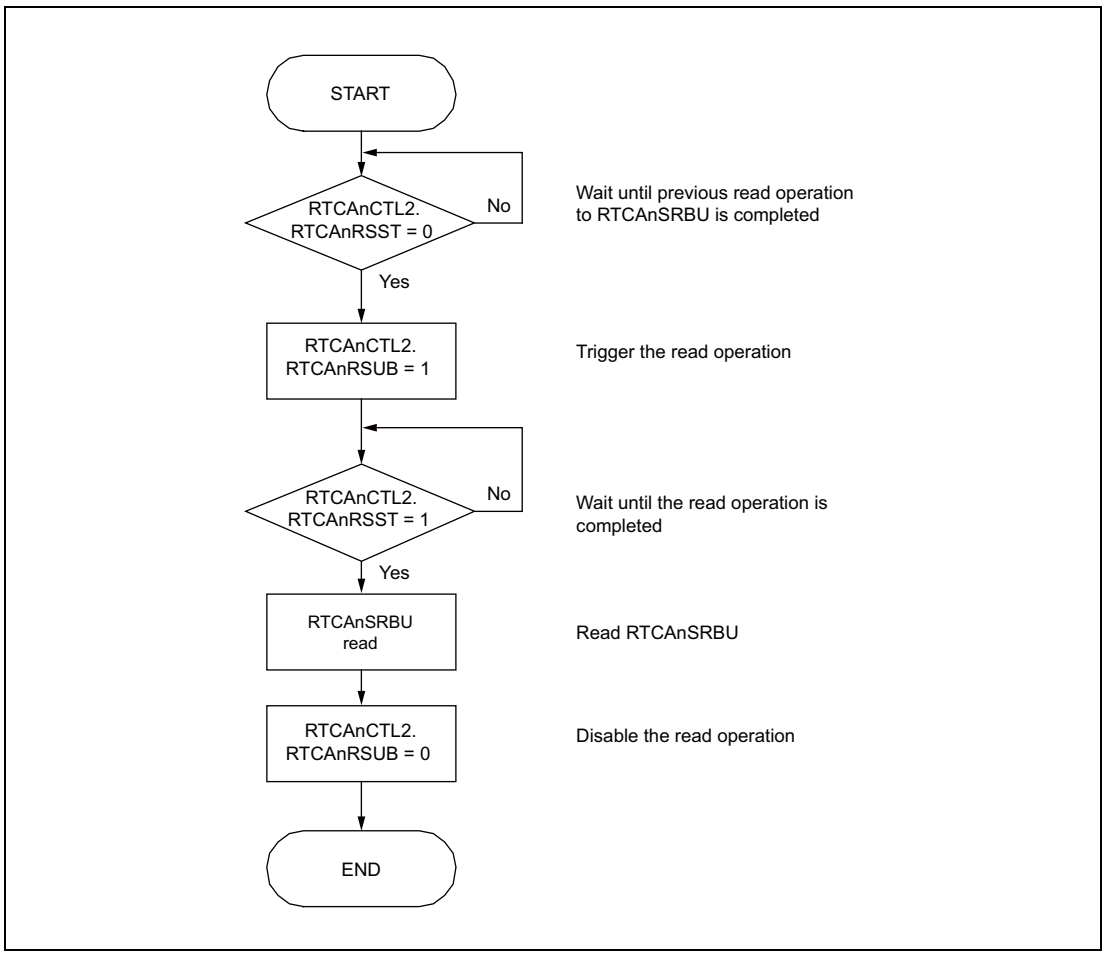



**Figure 27.9 Reading the RTCAnSRBU Register**

### 27.5.5 Writing to RTCAnSUBU

RTCAnSUBU is the clock error correction register for the sub-counter.

When the sub-counter operation is enabled (RTCAnCTL0.RTCAnCE = 1), write to RTCAnSUBU according to the flow described below.

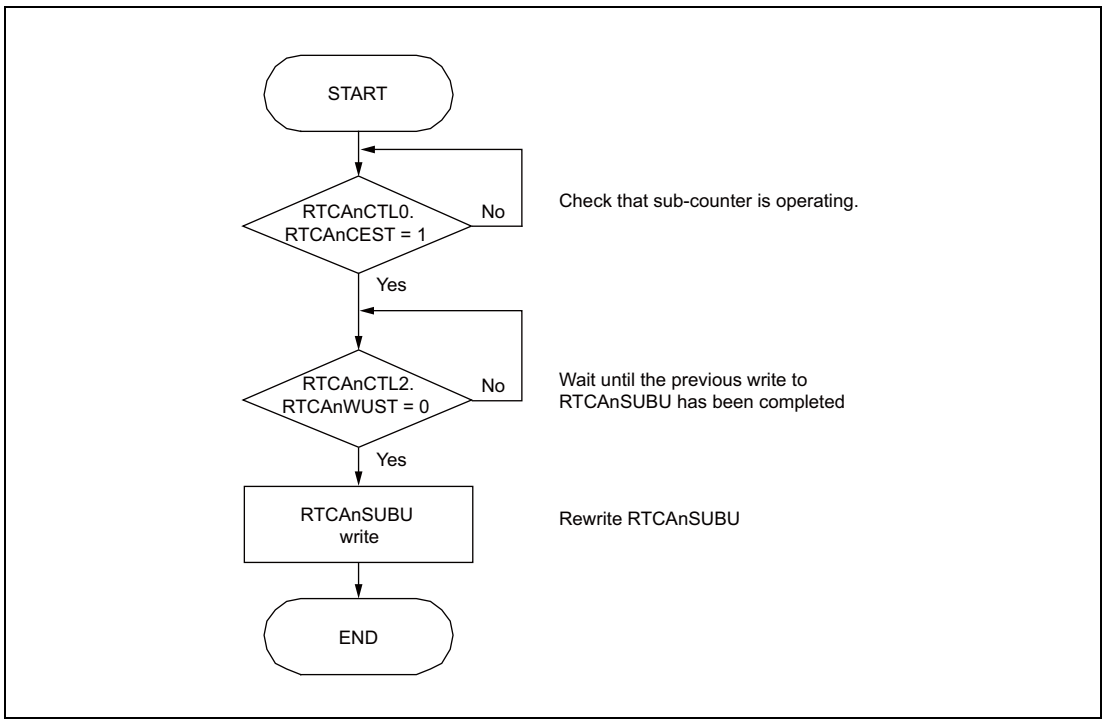



**Figure 27.10 Writing to the RTCAnSUBU Register**

**NOTE**

While the sub-counter operation is enabled (RTCAnCTL0.RTCAnCE = 1), the status flag RTCAnCTL2.RTCAnWUST is set when RTCAnSUBU is written to. It is cleared when the write operation to RTCAnSUBU is completed. This is synchronous with the next RTCAnSUBC overflow.
RTCAnCTL2.RTCAnWUST can be set for up to one second. Be careful when performing polling (checking if RTCAnCTL2.RTCAnWUST = 1 at the beginning of this flow).

### 27.5.6 Writing to RTCAnSCMP

RTCAnSCMP is the sub-counter compare register.

When the sub-counter operation is enabled (RTCAnCTL0.RTCAnCE = 1), write to RTCAnSCMP according to the flow described below.

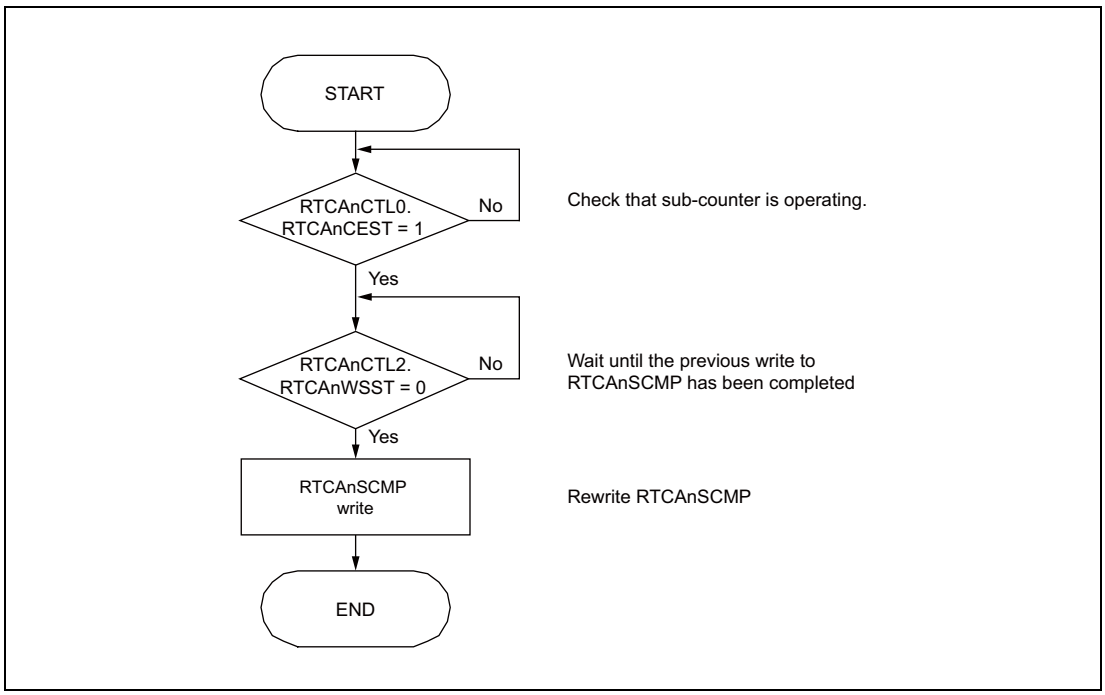



**Figure 27.11 Writing the RTCAnSCMP Register**

**NOTE**

While the sub-counter operation is enabled (RTCAnCTL0.RTCAnCE = 1), the status flag RTCAnCTL2.RTCAnWSST is set when RTCAnSCMP is written to. It is cleared when the write operation to RTCAnSCMP is completed. This is synchronous with the next RTCAnSUBC overflow.

RTCAnCTL2.RTCAnWSST can be set for up to one second. Be careful when performing polling (checking if RTCAnCTL2.RTCAnWSST = 1 at the beginning of this flow).

# 27.6 Timing Diagrams

## 27.6.1 Timing of Counter Start

The following diagram illustrates the counter start after setting the time in the buffer registers.

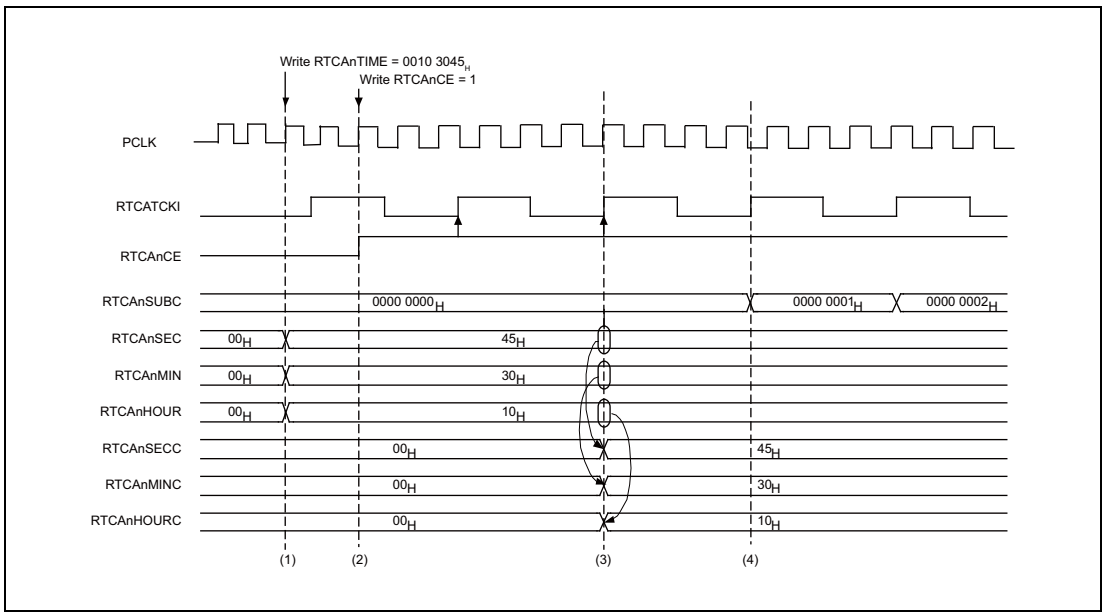



**Figure 27.12 Counter Start Timing**

The timing diagram above shows the following:

(1) The initial setting value of the time count buffer is set to 10:30:45 by setting RTCAnTIME = 0010 $3045_H$.
Count buffer registers RTCAnSEC, RTCAnMIN, and RTCAnHOUR are also automatically written.

(2) Sub-counter operation is started by setting RTCAnCTL0.RTCAnCE = 1.

(3) When the second rising edge of RTCATCKI occurs, the buffer register values are loaded to the corresponding count registers.

(4) When the next rising edge of RTCATCKI occurs, count up of the sub-counter starts.

### 27.6.2 Timing of Clock Counter Update while Counter Is Enabled

The following diagram illustrates the counter restart after setting the time in the buffer registers.

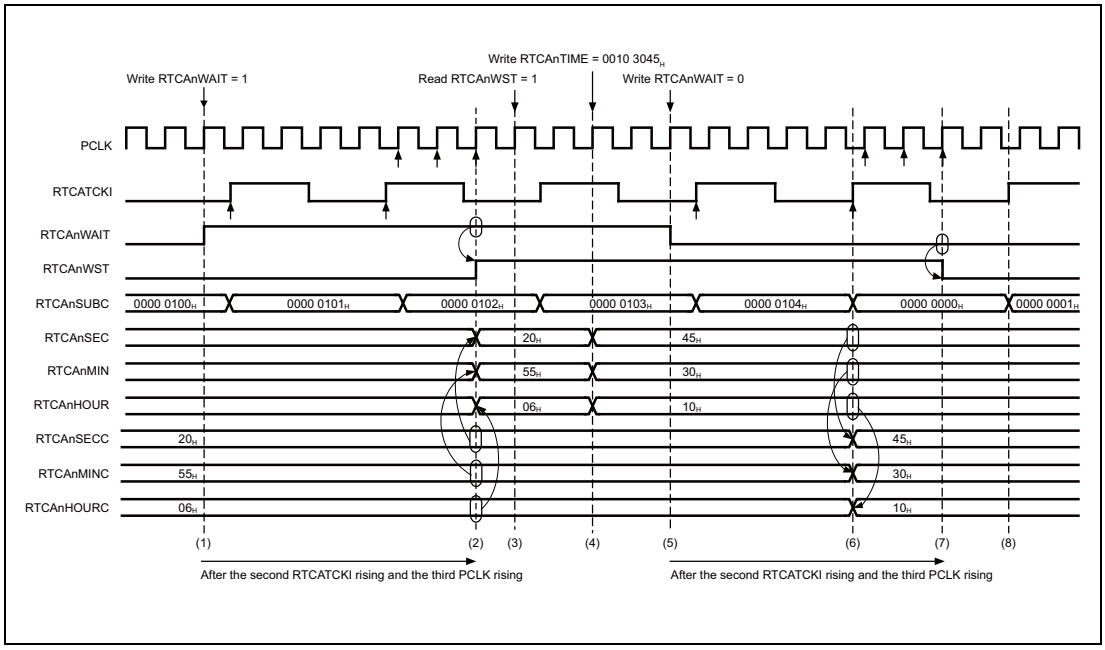



**Figure 27.13 Clock Counter Update Timing**

The timing diagram above shows the following:

(1) Trigger the clock counters stop (RTCAnCTL2.RTCAnWAIT = 1).

(2) RTCAnCTL2.RTCAnWST is set to 1 after the second rising edge of RTCATCKI and the third rising edge of PCLK, and the counter clock stops. The sub-counter continues counting.

(3) RTCAnCTL2.RTCAnWST = 1 can be readable.

(4) The initial setting value of the time count buffer is set to 10:30:45 by setting RTCAnTIME to 0010 $3045_H$.
Count buffer registers RTCAnSEC, RTCAnMIN, and RTCAnHOUR are also automatically written.

(5) Trigger the clock counters restart (RTCAnCTL2.RTCAnWAIT = 0).

(6) When the second rising edge of RTCATCKI occurs, the values of the buffer registers are loaded to the corresponding count registers. Write operation to RTCAnSECC is performed and RTCAnSUBC is cleared.

(7) When the third rising edge of PCLK occurs, RTCAnCTL2.RTCAnWST is set to 0.

(8) Clock counter operation is resumed.

### 27.6.3 Timing of Sub-Counter Read Buffer Reading while Counter is Enabled

The following diagram illustrates the timing when reading the sub-counter read buffer RTCAnSRBU.

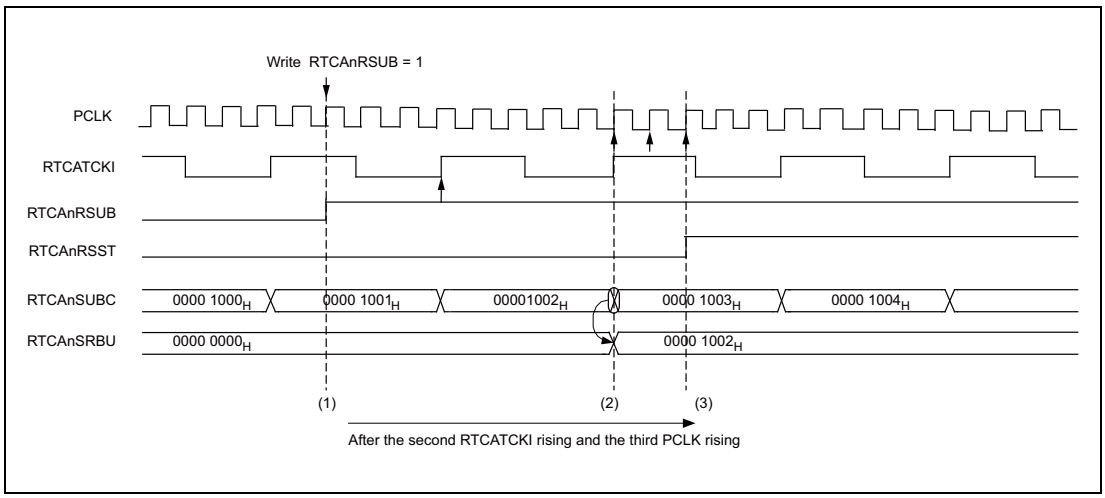



**Figure 27.14 Timing when Reading the Sub-Counter Read Buffer Register Value**

The timing diagram above shows the following:

(1) Setting RTCAnRSUB = 1 triggers loading of the sub-counter value to RTCAnSRBU.

(2) When the second rising edge of RTCATCKI occurs, the value of RTCAnSUBC is loaded to RTCAnSRBU.

(3) When the third rising edge of PCLK occurs, RTCAnCTL2.RTCAnRSST is set to 1 and RTCAnSRBU can be read.

# Section 28 Encoder Timer (ENCA)

This section contains a generic description of the Encoder Timer (ENCA).

The first part in this section describes all RH850/F1K specific properties, such as the number of units, register base addresses, etc.

The remainder of the section describes the functions and registers of the ENCA.

## 28.1 Features of RH850/F1K ENCA

### 28.1.1 Number of Units and Channels

This microcontroller has the following number of ENCA units.

Each ENCA unit has one channel ENCA. “Number of channels” is used with the same meaning as “number of units” in this section.

**Table 28.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of Units | 1 | | |
| Name | ENCAn (n = 0) | | |

**Table 28.2 Index**

| Index | Description |
|---|---|
| n | Throughout this section, the individual ENCA units are identified by the index “n” (n = 0); for example, ENCAnCTL is the ENCAn control register. |

### 28.1.2 Register Base Address

ENCAn base addresses are listed in the following table.

ENCAn register addresses are given as offsets from the base addresses.

**Table 28.3 Register Base Address**

| Base Address Name | Base Address |
|---|---|
| <ENCA0_base> | FFE8 $0000_H$ |

### 28.1.3 Clock Supply

The ENCAn clock supply is shown in the following table.

**Table 28.4 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| ENCA0 | PCLK | CKSCLK_IPERI1 | Module clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CKSCLK_IPERI1 | |

### 28.1.4 Interrupt Requests

ENCAn interrupt requests are listed in the following table.

**Table 28.5 Interrupt Requests**

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **ENCA0** | | | |
| ENCATIOV | Overflow interrupt | 85 | — |
| ENCATIUD | Underflow interrupt | 86 | — |
| ENCATINT0 | Capture/compare match interrupt 0 | 87 | — |
| ENCATINT1 | Capture/compare match interrupt 1 | 88 | — |
| ENCATIEC | Encoder clear interrupt | 89 | — |

### 28.1.5 Reset Sources

ENCAn reset sources are listed in the following table. ENCAn is initialized by these reset sources.

**Table 28.6 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| ENCA0 | All reset sources (ISORES) |

### 28.1.6 External Input/Output Signals

External input/output signals of ENCAn are listed below.

**Table 28.7 External Input/Output Signals**

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **ENCA0** | | |
| ENCATTIN0 | ENCAn capture trigger input 0*1 | ENCA0TIN0 |
| ENCATTIN1 | ENCAn capture trigger input 1*1 | ENCA0TIN1 |
| ENCAnE0 | ENCAn encoder input 0*1 | ENCA0E0 |
| ENCAnE1 | ENCAn encoder input 1*1 | ENCA0E1 |
| ENCAnEC | ENCAn encoder clear input*1 | ENCA0EC |

Note 1. When channel input pins are to be used, noise filters must be set for the corresponding port pin functions. For details, see **Section 2.12, Noise Filter & Edge/Level Detector**.

### 28.1.7 Internal Input/Output Signals

Input/output signals to be connected between ENCA and PIC are listed below.

**Table 28.8 Internal Input/Output Signals**

| Unit Signal Name | Description | Connected to |
|---|---|---|
| ENCATSST | Simultaneous start trigger | PIC |
| ENCATTIN1 | ENCAn capture trigger input 1 | PIC |

## 28.2 Overview

### 28.2.1 Functional Overview

- Generation of the counter control signal from the encoder input signal, and count operation in synchronization with PCLK
- Capture function for capturing the counter value with an external trigger signal
- Compare function for compare match judgment with the counter value
- Two capture/compare registers that can be set separately for capture operation and for compare operation
- Interrupt mask function for masking the interrupt request signal output as a result of compare match judgment during compare operation
- Function for loading the value of the capture/compare register to the counter upon underflow occurrence
- Encoder input signal can be used as the timer counter clear condition
- Edge or level can be selected for determining the presence of the encoder input signal that is used as the timer counter clear condition
- Detection of counter overflow and underflow and output of error flags and error occurrence interrupts
- Five interrupts: two capture/compare interrupts, one counter clear interrupt, one overflow interrupt, and one underflow interrupt

## 28.2.2 Block Diagram

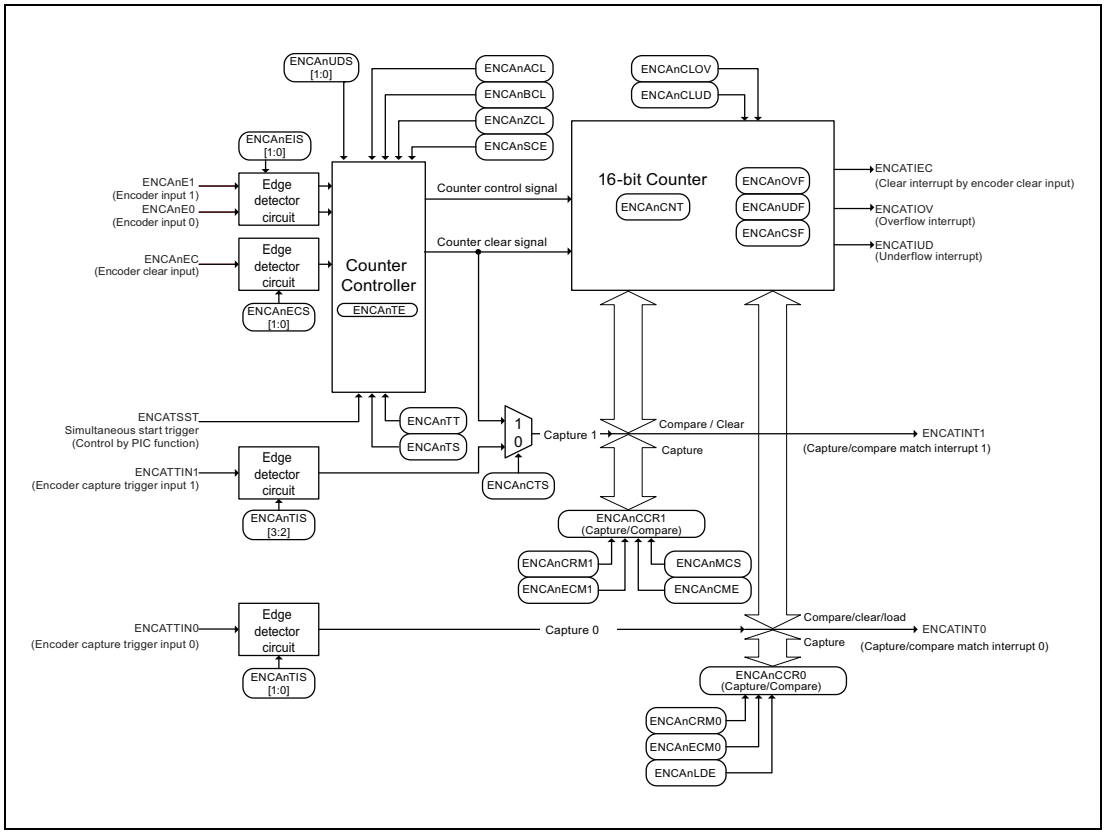



**Figure 28.1 ENCA Block Diagram**

# 28.3 Registers

## 28.3.1 List of Registers

ENCA registers are listed in the following table.

<ENCAn_base> is defined in **Section 28.1.2, Register Base Address**.

**Table 28.9 List of Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| ENCAn | ENCAn capture/compare register 0 | ENCAnCCR0 | <ENCAn_base> |
| | ENCAn capture/compare register 1 | ENCAnCCR1 | <ENCAn_base> + $04_H$ |
| | ENCAn counter register | ENCAnCNT | <ENCAn_base> + $08_H$ |
| | ENCAn status flag register | ENCAnFLG | <ENCAn_base> + $0C_H$ |
| | ENCAn status flag clear register | ENCAnFGC | <ENCAn_base> + $10_H$ |
| | ENCAn timer enable status register | ENCAnTE | <ENCAn_base> + $14_H$ |
| | ENCAn timer start trigger register | ENCAnTS | <ENCAn_base> + $18_H$ |
| | ENCAn timer stop trigger register | ENCAnTT | <ENCAn_base> + $1C_H$ |
| | ENCAn I/O control register 0 | ENCAnIOC0 | <ENCAn_base> + $20_H$ |
| | ENCAn control register | ENCAnCTL | <ENCAn_base> + $40_H$ |
| | ENCAn I/O control register 1 | ENCAnIOC1 | <ENCAn_base> + $44_H$ |
| | ENCAn emulation register | ENCAnEMU | <ENCAn_base> + $48_H$ |

## 28.3.2 ENCAnCTL — ENCAn Control Register

This register is used to configure various operation settings of the Encoder Timer.

**Access:** This register can be read or written in 16-bit units.
Writing to this register during operation is prohibited.

**Address:** <ENCAn_base> + $40_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ENCAn CME | ENCAn MCS | — | — | — | — | ENCAn CRM1 | ENCAn CRM0 | ENCAn CTS | — | — | ENCAn LDE | ENCAn ECM1 | ENCAn ECM0 | ENCAnUDS [1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 28.10 ENCAnCTL Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | ENCAnCME | Encoder Clear Mask Enable<br>This bit is used to enable/disable masking of compare-match interrupt detection when the compare function is used.<br>0: Disables the compare-match interrupt (ENCATINT1) mask function for the ENCAnCCR1 register<br>1: Enables the compare-match interrupt (ENCATINT1) mask function for the ENCAnCCR1 register.<br>This bit is valid only when ENCAnCRM1 = 0.<br>When this bit is set to “1”, setting ENCAnECM1 to “1” is prohibited. |
| 14 | ENCAnMCS | Encoder Mask Clear Select<br>This bit is used to select the trigger for cancelling masking of compare-match interrupt detection ENCATINT1 when the compare function is used.<br>This bit is valid only when ENCAnCRM1 = 0.<br>0: Masking of compare-match interrupt detection is canceled when the ENCAnCCR1 register is written.<br>1: Masking of compare match interrupt detection is canceled when one of the following three operations is performed.<br>– Timer counter clear operation accompanying encoder clear input<br>– Timer counter clear operation upon compare-match between ENCAnCNT and ENCAnCCR0 when ENCAnECM0 = 1<br>– Loading from ENCAnCCR0 to the timer counter upon underflow detection when ENCAnLDE = 1 |
| 13 to 10 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 9 | ENCAnCRM1 | ENCAnCCR1 Register Mode<br>0: ENCAnCCR1 used as compare register.<br>1: ENCAnCCR1 used as capture register. |
| 8 | ENCAnCRM0 | ENCAnCCR0 Register Mode<br>0: ENCAnCCR0 used as compare register.<br>1: ENCAnCCR0 used as capture register. |
| 7 | ENCAnCTS | ENCAnCCR1 Capture Trigger Select<br>This is a trigger selection bit for the capture operation to the ENCAnCCR1 register.<br>This bit is valid only when ENCAnCRM1 = 1.<br>0: Uses ENCATTIN1 of capture trigger 1 signal as the capture trigger for the ENCAnCCR1 register.<br>1: The counter clear signal selected with ENCAnSCE is used as the capture trigger for the ENCAnCCR1 register. |
| 6, 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

**Table 28.10 ENCAnCTL Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 4 | ENCAnLDE | ENCAn Counter Load Enable<br>This bit is used to enable/disable setting value loading to the counter upon underflow occurrence.<br>This bit is valid only when ENCAnCRM0 = 0.<br>When ENCAnCRM0 = 1, loading of the ENCAnCCR0 register setting value to the counter upon occurrence of an underflow is not performed, regardless of the value of this bit.<br>0: Disables loading of ENCAnCCR0 register setting value to counter upon occurrence of a counter underflow.<br>1: Enables loading of ENCAnCCR0 register setting value to counter upon occurrence of a counter underflow. |
| 3 | ENCAnECM1 | Encoder Clear Mode 1<br>This bit is used to set the counter clear operation upon match between the counter value and ENCAnCCR1 setting value.<br>This bit is valid only when ENCAnCRM1 = 0.<br>0: Does not clear the counter to $0000_H$ upon match of timer counter value and ENCAnCCR1 setting value.<br>1: Clears the counter to $0000_H$ upon match of timer counter value and ENCAnCCR1 setting value if the next count is a down-count. |
| 2 | ENCAnECM0 | Encoder Clear Mode 0<br>This bit is used to set the counter clear operation upon match between the counter value and ENCAnCCR0 setting value.<br>This bit is valid only when ENCAnCRM0 = 0.<br>0: Does not clear the counter to $0000_H$ upon match of timer counter value and ENCAnCCR0 setting value.<br>1: Clears the counter to $0000_H$ upon match of timer counter value and ENCAnCCR0 setting value if the next count is a up-count. |
| 1, 0 | ENCAnUDS[1:0] | Up/down Count Selection 1 and 0<br>These are the counter up/down control bits using ENCAnE0 and ENCAnE1.<br>00: Upon detection of valid edge of ENCAnE0,<br>- down-count when ENCAnE1 = H,<br>- up-count when ENCAnE1 = L<br>01: Upon detection of valid edge of ENCAnE0, up-count,<br>Upon detection of valid edge of ENCAnE1, down-count<br>10: At rising edge of ENCAnE0, down-count<br>At falling edge of ENCAnE0, up-count<br>However, count operation is performed only when ENCAnE1 = L.<br>11: Detection of both edges of ENCAnE0, ENCAnE1.<br>The count operation is determined based on the combination of the detected edge and level. |

### 28.3.3 ENCAnIOC0 — ENCAn I/O Control Register 0

This register is used to select the input edge of capture triggers 0 and 1 (ENCATTIN0, ENCATTIN1).

**Access:** This register can be read or written in 8-bit units.

**Address:** <ENCAn_base> + $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | ENCAnTIS[3:2] | | ENCAnTIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 28.11 ENCAnIOC0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3, 2 | ENCAnTIS[3:2] | Input Edge Selection for Capture Trigger 1<br>These bits are valid only when ENCAnCTL.ENCAnCRM1 = 1 and ENCAnCTL.ENCAnCTS = 0.<br>All other settings of ENCAnCRM1 and ENCAnCTS are invalid.<br>00: No edge detection<br>01: Rising edge detection<br>10: Falling edge detection<br>11: Both edges detection |
| 1, 0 | ENCAnTIS[1:0] | Input Edge Selection for Capture Trigger 0<br>These bits are valid only when ENCAnCTL.ENCAnCRM0 = 1.<br>00: No edge detection<br>01: Rising edge detection<br>10: Falling edge detection<br>11: Both edges detection |

### 28.3.4 ENCAnIOC1 — ENCAn I/O Control Register 1

This register is used to perform the clear condition setting and edge selection for input from the encoder.

**Access:** This register can be read or written in 8-bit units.
Writing to this register during operation is prohibited.

**Address:** <ENCAn_base> + $44_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ENCAnSCE | ENCAnZCL | ENCAnBCL | ENCAnACL | ENCAnECS[1:0] | | ENCAnEIS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 28.12 ENCAnIOC1 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | ENCAnSCE | Encoder Special-Clear Enable<br>This is an encoder special clear enable bit.<br>When setting this bit to 1, set ENCAnUDS1 and ENCAnUDS0 to $10_B$ or $11_B$. The operation is not guaranteed if this bit is set to 1 with ENCAnUDS1 and ENCAnUDS0 set to $00_B$ or $01_B$.<br>0: Clears the counter upon detection of ENCAnEC valid edge (set with ENCAnECS1 and ENCAnECS0).<br>1: Clears the counter upon detection of input level condition of ENCAnE0, ENCAnE1 and ENCAnEC (set with ENCAnZCL bit, ENCAnBCL bit, and ENCAnACL bit). |
| 6 | ENCAnZCL | Input-Z Clear Condition Selection<br>This bit is used to set the condition for clearing the encoder clear input (ENCAnEC) when using the encoder special clear function.<br>This bit is valid only when ENCAnSCE = 1; it is invalid when ENCAnSCE = 0.<br>0: Clear condition: Low level<br>1: Clear condition: High level |
| 5 | ENCAnBCL | Input-B Clear Condition Selection<br>This bit is used to set the condition for clearing the encoder input 1 (ENCAnE1) when using the encoder special clear function.<br>This bit is valid only when ENCAnSCE = 1; it is invalid when ENCAnSCE = 0.<br>0: Clear condition: Low level<br>1: Clear condition: High level |
| 4 | ENCAnACL | Input-A Clear Condition Selection<br>This bit is used to set the condition for clearing the encoder input 0 (ENCAnE0) when using the encoder special clear function.<br>This bit is valid only when ENCAnSCE = 1; it is invalid when ENCAnSCE = 0.<br>0: Clear condition: Low level<br>1: Clear condition: High level |
| 3, 2 | ENCAnECS[1:0] | Encoder Clear Input Edge Selection 1 and 0<br>These are the encoder clear input edge selection bits.<br>These bits are valid only when ENCAnSCE = 0; they are invalid when ENCAnSCE = 1.<br>00: No edge detection<br>01: Rising edge detection<br>10: Falling edge detection<br>11: Both edges detection |

**Table 28.12 ENCAnIOC1 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 1, 0 | ENCAnEIS[1:0] | Encoder Edge Input Selection 1 and 0<br>These are the encoder input edge selection bits.<br>These bits are valid when ENCAnUDS1 and ENCAnUDS0 = $00_B$ or $01_B$, and are invalid when ENCAnUDS1 and ENCAnUDS0 = $10_B$ or $11_B$.<br>00: No edge detection<br>01: Rising edge detection<br>10: Falling edge detection<br>11: Both edges detection |

### 28.3.5 ENCAnFLG — ENCAn Status Flag Register

This register holds the status flags of the timer counter of ENCAn.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <ENCAn_base> + $0C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | ENCAnCSF | ENCAnUDF | ENCAnOVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 28.13 ENCAnFLG Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 3 | Reserved | When read, the value after reset is returned. |
| 2 | ENCAnCSF | Counter Status Flag<br>This bit reflects the current timer counter operation.<br>0: Timer counter in up-count status<br>1: Timer counter in down-count status |
| 1 | ENCAnUDF | Underflow Flag<br>This bit reflects the occurrence of an underflow during the timer counter operation. This bit is cleared to 0 at the start of count operation.<br>0: This flag is cleared upon any of the following events:<br>– “1” is written to ENCAnFGC.ENCAnCLUD<br>– The flag is cleared to 0 by setting ENCAnTS bit to “1” when ENCAnTE = 0 or by setting the simultaneous start trigger input (ENCATSST signal) to “High”.<br>1: This flag is set to “1” upon occurrence of an underflow during the encoder timer count operation. |
| 0 | ENCAnOVF | Overflow Flag<br>This bit reflects the occurrence of an overflow during the timer counter operation.<br>This bit is cleared to 0 at the start of count operation.<br>0: This flag is cleared upon any of the following events:<br>– “1” is written to ENCAnFGC.ENCAnCLOV<br>– The flag is cleared to 0 by setting ENCAnTS bit to “1” when ENCAnTE = 0 or by setting the simultaneous start trigger input (ENCATSST signal) to “High”.<br>1: This flag is set to “1” upon occurrence of an overflow during the encoder timer count operation. |

### 28.3.6 ENCAnFGC — ENCAn Status Flag Clear Register

This register is used to clear the timer counter status flags of ENCAnFLG.

**Access:** This register is a write-only register that can be written in 8-bit units.
This register always returns 0 when read.

**Address:** <ENCAn_base> + $10_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | ENCAnCLUD | ENCAnCLOV |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | W | W |

**Table 28.14 ENCAnFGC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When writing, write the value after reset. |
| 1 | ENCAnCLUD | Underflow Flag Clear<br>This bit clears the underflow flag.<br>0: Writing is ignored.<br>1: Clears ENCAnUDF of the ENCAnFLG register (clears underflow detection). |
| 0 | ENCAnCLOV | Overflow Flag Clear<br>This bit clears the overflow flag.<br>0: Writing is ignored.<br>1: Clears ENCAnOVF of the ENCAnFLG register (clears overflow detection). |

### 28.3.7 ENCAnCCR0 — ENCAn Capture/Compare Register 0

This register is a 16-bit capture/compare register 0.

**Access:** This register can be read or written in 16-bit units.
When this register functions as a capture register, only reading is possible. Write operation is ignored.
When this register functions as a compare register, reading and writing is possible.

**Address:** <ENCAn_base> + $00_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ENCAnCCR0[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 28.15 ENCAnCCR0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | ENCAnCCR0 [15:0] | Capture/Compare Register 0<br>Upon occurrence of an underflow, the setting value of this register may be loaded to the counter according to the ENCAnCTL.ENCAnLDE setting. See the description of the ENCAnLDE bit in ENCA control register ENCAnCTL for details.<br>• If ENCAnCTL.ENCAnCRM0 = 0: ENCAnCCR0 is a compare register. Set the value to be compared with the timer counter value.<br>• If ENCAnCTL.ENCAnCRM0 = 1: ENCAnCCR0 is a capture register. The captured timer counter value is stored. |

## 28.3.8 ENCAnCCR1 — ENCAn Capture/Compare Register 1

This register is a 16-bit capture/compare register 1.

**Access:** This register can be read or written in 16-bit units.
When this register functions as a capture register, only reading is possible. Write operation is ignored.
When this register functions as a compare register, reading and writing is possible.

**Address:** <ENCAn_base> + $04_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ENCAnCCR1[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 28.16 ENCAnCCR1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | ENCAnCCR1 [15:0] | Capture/Compare Register 1<br>During capture operation, the capture trigger to this register differs according to the ENCAnCTL.ENCAnCTS setting. See the description of the ENCAnCTS bit in ENCA control register ENCAnCTL for details.<br>• If ENCAnCTL.ENCAnCRM1 = 0: ENCAnCCR1 is a compare register. Set the value to be compared with the timer counter value.<br>• If ENCAnCTL.ENCAnCRM1 = 1: ENCAnCCR1 is a capture register. The captured timer counter value is stored. |

## 28.3.9 ENCAnCNT — ENCAn Counter Register

This register is the 16-bit timer counter register.

**Access:** This register can be read or written in 16-bit units.
This register can be written only when the operation is stopped.

**Address:** <ENCAn_base> + $08_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ENCAnCNT[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 28.17 ENCAnCNT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | ENCAnCNT [15:0] | Counter Register<br>• ENCAnTE.ENCAnTE status: 0 (initial setting): Count stop<br>An arbitrary value can be set to timer counter.<br>• ENCAnTE.ENCAnTE status: 0 → 1 (operation start): Count operation start<br>Up/down count operation is started with the set arbitrary value.<br>• ENCAnTE.ENCAnTE status: 1 (operating): Counting<br>Up/down count operation is performed.<br>• ENCAnTE.ENCAnTE status: 1 → 0 (stopped): Count stop<br>The counter value immediately before the operation was stopped is held, and the count operation is stopped. |

## 28.3.10 ENCAnTE — ENCAn Timer Enable Status Register

This register indicates the operating status of ENCAn.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <ENCAn_base> + $14_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | ENCAnTE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 28.18 ENCAnTE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | ENCAnTE | Timer Status Enable<br>This is a status bit that indicates the operation enabled/stopped status of ENCAn.<br>This bit is cleared to 0 when “1” is written to ENCAnTT.ENCAnTT.<br>This bit is set to “1” when “1” is written to ENCAnTS.ENCAnTS, or when the input signal of ENCATSST is set to High level.<br>0: Operation stopped status<br>1: Operation enabled status |

### 28.3.11 ENCAnTS — ENCAn Timer Start Trigger Register

This register provides the trigger bit for setting the ENCAn to the operation enabled state.

**Access:** This register is a write-only register that can be written in 8-bit units.
It is always read as $00_H$. This register can be written only when ENCAnTE.ENCAnTE is 0.

**Address:** <ENCAn_base> + $18_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | ENCAnTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 28.19 ENCAnTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | ENCAnTS | Timer Start Trigger<br>This is the trigger bit that sets the ENCAn to the operation enabled state.<br>0: Writing is ignored.<br>1: The ENCAn is set to the operation enabled state by setting ENCAnTE.ENCAnTE = 1. |

## 28.3.12 ENCAnTT — ENCAn Timer Stop Trigger Register

This register provides the trigger bit for setting the ENCAn to the operation stopped state.

**Access:** This register is a write-only register that can be written in 8-bit units.
It is always read as $00_H$.

**Address:** <ENCAn_base> + $1C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | ENCAnTT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 28.20 ENCAnTT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | ENCAnTT | Timer Stop Trigger<br>This is the trigger bit that sets the ENCAn to the operation stopped state.<br>0: Writing is ignored.<br>1: Clears ENCAnTE.ENCAnTE to “0”, to set the ENCAn to the count operation stopped state. |

## 28.3.13 ENCAnEMU — ENCAn Emulation Register

This register controls operations by SVSTOP.

**Access:** This register can be read or written in 8-bit units.
Writing to this register should be performed in the counter operation stopped status (ENCAnTE.ENCAnTE = 0 and EPC.SVSTOP = 0).

**Address:** <ENCAn_base> + $48_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ENCAnSVSDIS | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R |

**Table 28.21 ENCAnEMU Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | ENCAnSVSDIS | • When EPC.SVSTOP bit = 0:<br>The count clock continues to be provided when the debugger assumes control of the microcontroller (at a break point, etc.), regardless of the value of this bit (1 or 0).<br>• When EPC.SVSTOP bit = 1:<br>0: The count clock is stopped when the debugger assumes control of the microcontroller (at a break point, etc.).<br>1: The count clock continues to be provided when the debugger assumes control of the microcontroller (at a break point, etc.). |
| 6 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

## 28.4 Operation

The ENCAn operates the timer counter with counter up/down control and clear control by encoder inputs. The ENCAnCCR0 and ENCAnCCR1 registers can be used as dedicated compare registers or as dedicated capture registers.

### 28.4.1 Timer Counter Operation

The timer counter operations of the ENCAn are described below.

The figure below shows the operation phases. See the corresponding section with the section number for detailed descriptions on each operation.

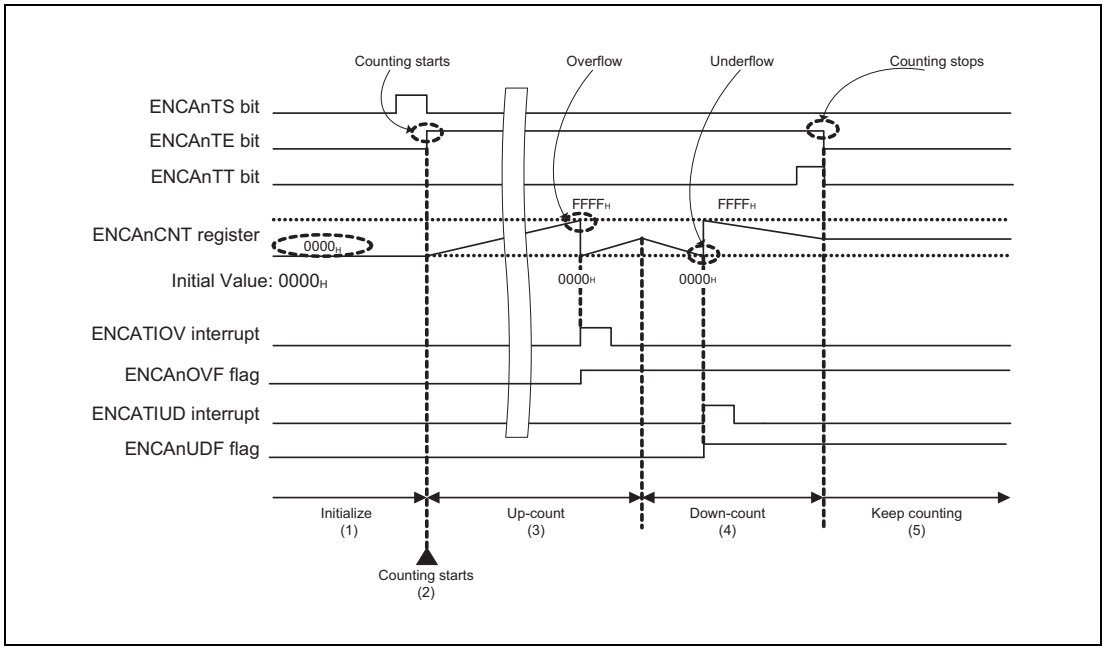



**Figure 28.2 Timer Counter Initial Value Setting/Start/Stop**

#### (1) Timer Counter Initial Value Setting

The initial value of the ENCAn counter register (ENCAnCNT) can be set in the counter operation stopped status (ENCAnTE = 0).

#### (2) Timer Counter Startup

By writing "1" to the timer start trigger bit (ENCAnTS), the timer status enable bit (ENCAnTE) is set to "1", the count operation is enabled, and counting operation is performed upon detection of the valid edge of the encoder input.

#### (3) Overflow Operation

An overflow occurs when up-counting is performed when the counter value is $FFFF_H$. If the counter value changes from $FFFF_H$ to $0000_H$, an overflow interrupt (ENCATIOV) is generated, and the overflow flag (ENCAnOVF) is set to "1". The overflow flag (ENCAnOVF) is cleared to "0" when "1" is set to the overflow flag clear bit (ENCAnCLOV). For details about the operation, see **Section 28.6.1, Overflow Occurrence and Overflow Flag Clear Operation**.

#### (4) Underflow Operation

An underflow occurs when down-counting is performed when the counter value is $0000_H$. If the counter value changes from $0000_H$ to $FFFF_H$, an underflow interrupt (ENCATIUD) is generated, and the underflow flag (ENCAnUDF) is set to “1”. The underflow flag (ENCAnUDF) is cleared to “0” when “1” is set to the underflow flag clear bit (ENCAnCLUD). For details about the operation, see **Section 28.6.2, Underflow Occurrence and Underflow Flag Clear Operation**.

#### (5) Timer Counter Stop

By writing “1” to the timer stop trigger bit (ENCAnTT), the timer status enable bit (ENCAnTE) is cleared to “0”, and the count operation is stopped. At this time, the timer counter is not reset to $0000_H$ and holds the value before count operation stop.

## 28.4.2 Up/Down Control of Timer Counter

Up/down control is performed by judging the phase of the encoder inputs (ENCAnE0, ENCAnE1) according to the settings of ENCAnUDS1 and ENCAnUDS0.

### 28.4.2.1 When the ENCAnUDS1, ENCAnUDS0 Bits in the ENCAnCTL Register = $00_B$

Table 28.22 When ENCAnUDS1, ENCAnUDS0 Bits = $00_B$

| ENCAnUDS1 | ENCAnUDS0 | Operation Description | | |
|---|---|---|---|---|
| | | ENCAnE0 Pin | ENCAnE1 Pin | Count Operation |
| 0 | 0 | Rising edge | High level | Down |
| | | Falling edge | | |
| | | Rising and falling edges | | |
| | | Rising edge | Low level | Up |
| | | Falling edge | | |
| | | Rising and falling edges | | |

The valid edge for ENCAnE0 is specified by setting ENCAnEIS1 and ENCAnEIS0.

Up/down count operation is performed when the valid edges and levels of ENCAnE0 and ENCAnE1 match.

The following timing chart shows the count operation when ENCAnUDS1, ENCAnUDS0 bits = $00_B$.

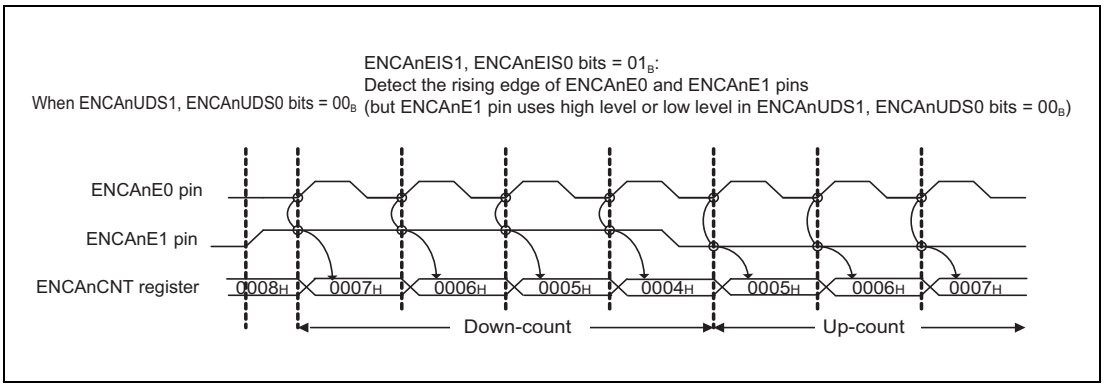



Figure 28.3 Count Operation when the ENCAnUDS1, ENCAnUDS0 Bits in the ENCAnCTL Register = $00_B$

### 28.4.2.2 When the ENCAnUDS1, ENCAnUDS0 Bits in the ENCAnCTL Register = $01_B$

Table 28.23 When the ENCAnUDS1, ENCAnUDS0 Bits = $01_B$

| ENCAnUDS1 | ENCAnUDS0 | Operation Description | | |
|---|---|---|---|---|
| | | ENCAnE0 Pin | ENCAnE1 Pin | Count Operation |
| 0 | 1 | Low level | Rising edge | Down |
| | | | Falling edge | |
| | | | Rising and falling edges | |
| | | High level | Rising edge | |
| | | | Falling edge | |
| | | | Rising and falling edges | |
| | | Rising edge | Low level | Up |
| | | Falling edge | | |
| | | Rising and falling edges | | |
| | | Rising edge | High level | |
| | | Falling edge | | |
| | | Rising and falling edges | | |
| | | Simultaneous input | | Hold |

The valid edges for ENCAnE0 and ENCAnE1 are specified by setting ENCAnEIS1 and ENCAnEIS0.

Up/down count operation is performed when the valid edges and levels of the ENCAnE0/ENCAnE1 pins match, and the count is held when the valid edges overlap.

The following timing chart shows the count operation when ENCAnUDS1, ENCAnUDS0 bits = $01_B$.

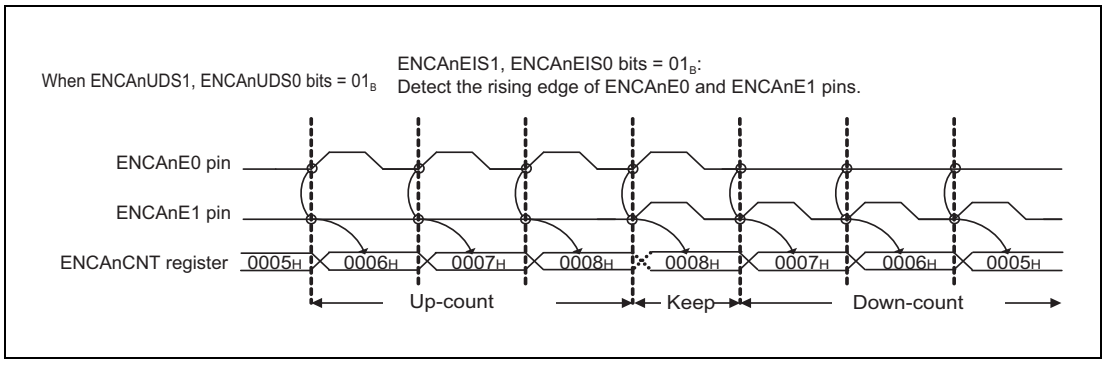



Figure 28.4 Count Operation when the ENCAnUDS1, ENCAnUDS0 Bits in the ENCAnCTL Register = $01_B$

### 28.4.2.3 When the ENCAnUDS1, ENCAnUDS0 Bits in the ENCAnCTL Register = $10_B$

Table 28.24 When the ENCAnUDS1, ENCAnUDS0 Bits = $10_B$

| ENCAnUDS1 | ENCAnUDS0 | Operation Description | | |
|---|---|---|---|---|
| | | ENCAnE0 Pin | ENCAnE1 Pin | Count Operation |
| 1 | 0 | Rising edge | Low level | Down |
| | | Rising edge | Falling edge | |
| | | Falling edge | Low level | Up |
| | | Falling edge | Falling edge | |
| | | Low level | Rising edge | Hold |
| | | Rising edge | Rising edge | |
| | | High level | Rising edge | |
| | | Falling edge | Rising edge | |
| | | Low level | Falling edge | |
| | | Rising edge | High level | |
| | | High level | Falling edge | |
| | | Falling edge | High level | |

The valid edge specification for ENCAnE0 and ENCAnE1 (settings of ENCAnEIS1 and ENCAnEIS0) is invalid.

The following timing chart shows the count operation when the ENCAnUDS1, ENCAnUDS0 bits = $10_B$.

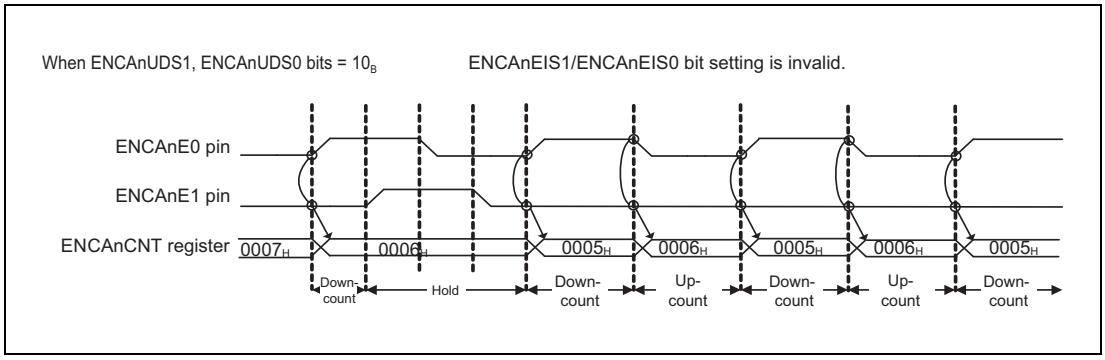



Figure 28.5 Count Operation when ENCAnUDS1, ENCAnUDS0 Bits in ENCAnCTL Register = $10_B$

### 28.4.2.4 When ENCAnUDS1, ENCAnUDS0 Bits in the ENCAnCTL Register = $11_B$

Table 28.25 When ENCAnUDS1, ENCAnUDS0 Bits = $11_B$

| ENCAnUDS1 | ENCAnUDS0 | Operation Description | | |
|---|---|---|---|---|
| | | ENCAnE0 Pin | ENCAnE1 Pin | Count Operation |
| 1 | 1 | Low level | Falling edge | Down |
| | | Rising edge | Low level | |
| | | High level | Rising edge | |
| | | Falling edge | High level | |
| | | Rising edge | High level | Up |
| | | High level | Falling edge | |
| | | Falling edge | Low level | |
| | | Low level | Rising edge | |
| | | Simultaneous input | | Hold |

Valid edge specification for ENCAnE0 and ENCAnE1 (settings of ENCAnEIS1 and ENCAnEIS0) is invalid.

The counter value is held when the valid edges of ENCAnE0 and ENCAnE1 overlap.

The following timing chart shows the count operation when ENCAnUDS1, ENCAnUDS0 bits = $11_B$.

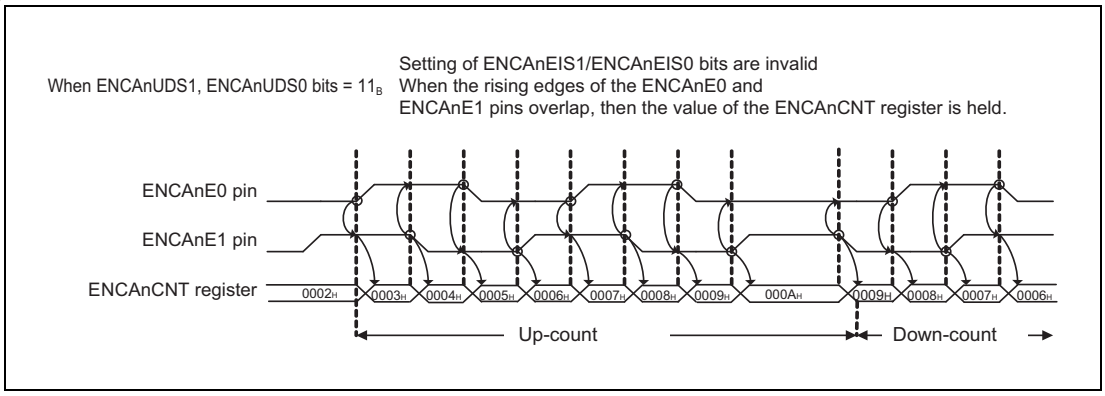



Figure 28.6 Count Operation when ENCAnUDS1, ENCAnUDS0 Bits in the ENCAnCTL Register = $11_B$

### 28.4.3 Timer Counter Clear Control by Encoder Input

The timer counter is cleared to $0000_H$ by encoder clear input (ENCAnEC).

Two types of clearing methods can be selected by controlling the ENCAnSCE, ENCAnZCL, ENCAnBCL, ENCAnACL, ENCAnECS1, and ENCAnECS0 bits of the ENCAnIOC1 register.

**Table 28.26 Timer Counter Clear Control by Encoder Input**

| Clearing method | ENCAnSCE | ENCAnZCL | ENCAnBCL | ENCAnACL | ENCAnECS1, ENCAnECS0 |
|---|---|---|---|---|---|
| (1) | 0 | Invalid | Invalid | Invalid | Valid |
| (2) | 1 | Valid | Valid | Valid | Invalid |

#### 28.4.3.1 Clearing Method when ENCAnSCE = 0

- Upon detection of the valid edge of ENCAnEC, the timer counter is cleared to $0000_H$ in synchronization with the operation clock.
- The valid edge of ENCAnEC is specified by the setting of the ENCAnECS1 and ENCAnECS0 bits.
- The settings of the ENCAnZCL, ENCAnBCL, and ENCAnACL bits are invalid.
- An encoder clear interrupt request signal (ENCATIEC) is output simultaneously with timer counter clearing.

For details about clear operation when ENCAnSCE = 0, see the timing chart in **Section 28.6.19, Capture Operation Performed upon Clearing by ENCAnEC when ENCAnSCE = 0**.

#### 28.4.3.2 Clearing Method when ENCAnSCE = 1

- When the clear levels of the ENCAnEC, ENCAnE1, ENCAnE0 inputs are detected, the timer counter is cleared to $0000_H$ in synchronization with the operating clock.
- Specify the clear levels of the ENCAnEC, ENCAnE1, ENCAnE0 inputs by the settings of the ENCAnZCL, ENCAnBCL, and ENCAnACL bits.
- The settings of the ENCAnECS1 and ENCAnECS0 bits are invalid.
- An encoder clear interrupt request signal (ENCATIEC) is output simultaneously with timer counter clearing.

The clearing conditions of the timer counter according to the ENCAnZCL, ENCAnBCL, and ENCAnACL settings are listed in the table below.

**Table 28.27 Clearing Conditions of the Timer Counter**

| Counter Clear Condition Setting | | | Encoder Pin Input Level | | |
|---|---|---|---|---|---|
| ENCAnZCL | ENCAnBCL | ENCAnACL | ENCAnEC | ENCAnE1 | ENCAnE0 |
| 0 | 0 | 0 | Low | Low | Low |
| 0 | 0 | 1 | Low | Low | High |
| 0 | 1 | 0 | Low | High | Low |
| 0 | 1 | 1 | Low | High | High |
| 1 | 0 | 0 | High | Low | Low |
| 1 | 0 | 1 | High | Low | High |
| 1 | 1 | 0 | High | High | Low |
| 1 | 1 | 1 | High | High | High |

## 28.4.4 Functions of ENCAnCCR0

### 28.4.4.1 Compare Function

- When ENCAnCRM0 = 0, the ENCAnCCR0 register functions as a dedicated compare register.
- Upon compare match between the value of the timer counter and the ENCAnCCR0 setting value, a compare 0 match interrupt (ENCATINT0) is output.
- When ENCAnECM0 = 1, the timer counter is cleared to $0000_H$ in synchronization with the operating clock upon compare match if the next count operation is up-count.

**Table 28.28 Compare Function of ENCAnCCR0**

| ENCAnCCR0 Function | Compare Match Clear Control | Next Count Operation | Timer Counter Clearing Upon Compare Match with ENCAnCCR0 |
|---|---|---|---|
| ENCAnCRM0 | ENCAnECM0 | | |
| 0 (Compare) | 0 | Up-count | Does not clear (continues count operation). |
| | | Down-count | |
| | 1 | Up-count | Clears timer counter to $0000_H$. |
| | | Down-count | Does not clear (continues count operation). |

**When ENCAnLDE = 1**

- Upon occurrence of an underflow, the setting value of the ENCAnCCR0 register is loaded to the timer counter.
- An underflow interrupt (ENCATIUD) is output.

**NOTE**

For the timing chart when ENCAnLDE = 1, see **Section 28.6.8, Using the ENCAnLDE Function Immediately after Startup** to **Section 28.6.12, Up-count after Conflict between ENCAnLDE Function (Loading Counter Value) and Clear Operation by Encoder Clear Input**.

### 28.4.4.2 Capture Function

- When ENCAnCRM0 = 1, the ENCAnCCR0 register functions as a dedicated capture register.
- Upon valid edge detection of the capture trigger input 0 (ENCATTIN0), the value of the timer counter is stored into ENCAnCCR0.
- A capture 0 interrupt (ENCATINT0) is output during capture operation.

**NOTE**

For details about capture operation for ENCAnCCR0, see the timing charts in **Section 28.6.14, Capture Operation between Count Clocks (ENCAnCCR0)** and **Section 28.6.17, Encoder Operation when Compare Match Clear Control is Disabled**.

## 28.4.5 Functions of ENCAnCCR1

### 28.4.5.1 Compare Function

- When ENCAnCRM1 = 0, the ENCAnCCR1 register functions as a dedicated compare register.
- Upon compare match between the value of the timer counter and the ENCAnCCR1 setting value, a compare 1 match interrupt (ENCATINT1) is output.
- When ENCAnECM1 = 1, the timer counter is cleared to $0000_H$ in synchronization with the operating clock upon compare match if the next count operation is down-count.

**Table 28.29 Compare Function of ENCAnCCR1**

| ENCAnCCR1 Function | Compare Match Clear Control | Next Count Operation | Timer Counter Clearing Upon Compare Match with ENCAnCCR1 |
|---|---|---|---|
| ENCAnCRM1 | ENCAnECM1 | | |
| 0 (Compare) | 0 | Up-count | Does not clear (continues count operation). |
| | | Down-count | |
| | 1 | Up-count | Does not clear (continues count operation). |
| | | Down-count | Clears timer counter to $0000_H$. |

#### Compare match interrupt mask function

- When ENCAnCME = 1, the compare 1 match interrupt mask function is enabled. In this state, the compare 1 match interrupt is output upon the first match of the value of the timer counter and the ENCAnCCR1 setting value, and interrupts are then masked for the second and subsequent compare matches.
- When ENCAnCME = 1 and ENCAnMCS = 0, a compare 1 match interrupt is output once upon the first compare match by writing to the ENCAnCCR1 register (interrupts are masked for the second and subsequent matches until the cancel trigger occurs again).
- When ENCAnCME = 1 and ENCAnMCS = 1, a compare 1 match interrupt is output once upon the first compare match by a timer counter clear operation accompanying encoder clear input or by a timer counter clear operation upon match between the ENCAnCCR0 register value and the timer counter value (interrupts are masked for the second and subsequent matches until the cancel trigger occurs again).
- When ENCAnCME = 1, ENCAnMCS = 1 and ENCAnLDE = 1, a compare 1 match interrupt is output once upon the first compare match by a loading operation of the ENCAnCCR0 register to the timer counter upon underflow detection (interrupts are masked for the second and subsequent matches until the cancel trigger occurs again).
- Setting ENCAnECM1 to “1” is prohibited when enabling the compare 1 match interrupt mask function.

Table 28.30 Compare Match Interrupt Mask Function of ENCAnCCR1

| ENCAnCCR1 Function | Compare 1 Match Interrupt Mask | Interrupt Mask Cancel Trigger | Compare 1 Match Interrupt Output upon Compare Match with ENCAnCCR1 |
|---|---|---|---|
| ENCAnCRM1 | ENCAnCME | ENCAnMCS | |
| 0 (Compare) | 0 (Mask function disabled) | — (Setting invalid) | Outputs compare 1 match interrupt upon each compare match. |
| | 1 (Mask function enabled) | 0 (Write operation to ENCAnCCR1) | Outputs compare 1 match interrupt once upon the first compare match. (Interrupts are masked for the second and subsequent matches until the cancel trigger occurs.) |
| | | 1 (Timer counter clear operation) (Loading from ENCAnCCR0 to the timer counter upon underflow occurrence when ENCAnLDE = 1) | |

### 28.4.5.2 Capture Function

When ENCAnCRM1 = 1, the ENCAnCCR1 register functions as a dedicated capture register.

**NOTE**

For details about capture operation to ENCAnCCR1, see the timing chart in **Section 28.6.13, Capture Operation between Count Clocks (ENCAnCCR1)**.

The operations for each of the ENCAnCTS settings are shown in the table below.

Table 28.31 Operations for Each of the ENCAnCTS Settings

| ENCAnCCR1 Function | Capture Trigger Selection | Capture Trigger Signal | Timer Counter Clearing | Interrupt Occurrence |
|---|---|---|---|---|
| ENCAnCRM1 | ENCAnCTS | | | |
| 1 (Capture) | 0 | Capture trigger 1 input (ENCATTIN1) | Does not clear timer counter. | (1) Capture 1 interrupt (ENCATINT1) |
| | 1 | Encoder clear input (set with ENCAnSCE) | Clears timer counter. | (1) Capture 1 interrupt (ENCATINT1)<br>(2) Encoder clear interrupt (ENCATIEC) |

**NOTE**

For details about the timing chart when ENCAnCTS = 0 or ENCAnCTS = 1, see the following:

**Section 28.6.3, Count Clearing and Capture Operation by Encoder Clear Input (ENCAnEC Pin)**, **Section 28.6.4, Conflict between Overflow Occurrence and Clear Operation by Encoder Clear Input (ENCAnEC Pin)**, **Section 28.6.5, Conflict between Underflow Occurrence and Clear Operation by Encoder Clear Input (ENCAnEC Pin)**, **Section 28.6.11, Conflict between ENCAnLDE Function (Loading Counter Value) and Clear Operation by Encoder Clear Input (ENCAnEC Pin)** and **Section 28.6.12, Up-count after Conflict between ENCAnLDE Function (Loading Counter Value) and Clear Operation by Encoder Clear Input**.

#### 28.4.5.3 Timer Counter Clearing upon Compare Register Match

Timer counter clearing upon compare match between the value of the timer counter and the ENCAnCCR0/1 setting value, according to the settings of the ENCAnECM1 and ENCAnECM0 bits in the ENCAnCTL register, is detailed in the following table.

**Table 28.32 Timer Counter Clearing Operation upon Compare Register Match**

| ENCAnECM1 and ENCAnECM0 | Next Count Operation | Timer Counter Clearing upon Compare Match with ENCAnCCR1 | Timer Counter Clearing upon Compare Match with ENCAnCCR0 |
|---|---|---|---|
| 00 | Up-count | Does not clear (continues count operation). | Does not clear (continues count operation). |
| | Down-count | Does not clear (continues count operation). | Does not clear (continues count operation). |
| 01 | Up-count | Does not clear (continues count operation). | Clears timer counter to $0000_H$. |
| | Down-count | Does not clear (continues count operation). | Does not clear (continues count operation). |
| 10 | Up-count | Does not clear (continues count operation). | Does not clear (continues count operation). |
| | Down-count | Clears timer counter to $0000_H$. | Does not clear (continues count operation). |
| 11 | Up-count | Does not clear (continues count operation). | Clears timer counter to $0000_H$. |
| | Down-count | Clears timer counter to $0000_H$. | Does not clear (continues count operation). |

### 28.4.6 Startup/Stop of Timer Counter

#### 28.4.6.1 Startup of Timer

The timer operation can be started by setting the ENCAnTS bit to “1”.

PIC setting enables simultaneous start with other timers. For details, see Section 29.8, Simultaneous Start Trigger Function.

#### 28.4.6.2 Stop of Timer

When the ENCAnTT bit is set to “1”, the ENCAnTE bit becomes “0” and the timer stops.

# 28.5 ENCAn Setting Sequences

## 28.5.1 ENCAn Setting Procedure

The ENCAn setting procedure is described below.

**Table 28.33 ENCAn Setting Procedure**

| Initial Setting | Action | Setting status |
|---|---|---|
| Initial setting | Reset deassertion | Power-on status, operation stopped status. (Writing to each register is enabled) |
| ENCAn initial setting | Perform the following initial settings.<br>• Setting for counter<br>• Setting for counter clear<br>• Setting for ENCAnCCR0 register<br>• Setting for ENCAnCCR1 register | This is the count operation stopped status. The value of the ENCAnTE bit indicating the operating status is 0. |
| | Perform the counter initial value settings.<br>• Set any 16-bit value to ENCAnCNT register. (When, after setting this register, the ENCAnTS bit is set to “1”, the counter operation starts from the set count value.) | The set value is set as the initial value of the counter register. |
| Operation start | Perform the counter operation start setting.<br>• Set the ENCAnTS bit to “1”. | This is the counter operation starts status. The value of the ENCAnTE bit indicating the operating status is 1, and the count clock is supplied to the internal circuit. |
| Operating | Only those registers whose setting can be changed during operation can be rewritten.<br>• ENCAnCCR0 register setting.<br>• ENCAnCCR1 register setting.<br>• ENCAnIOC0 register setting. | The count operation set with the initial setting is performed, and up/down counting is performed according to ENCAnE0 and ENCAnE1 pins. |
| Operation stop | Perform the counter operation stop setting during operation.<br>• Set the ENCAnTT bit to “1”. | This is the counter operation stopped status. The value of the ENCAnTE bit indicating the operating status is 0. |
| ENCAn stop | Reset | The setting registers are initialized. |

### 28.5.1.1 Initial Setting Procedure for the Counter

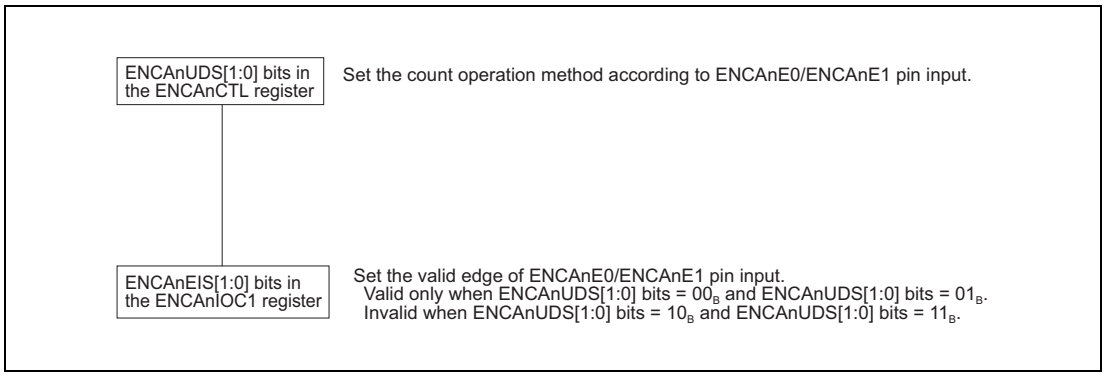



**Figure 28.7 Initial Setting Procedure for the Counter**

### 28.5.1.2 Initial Setting Procedure for Counter Clear

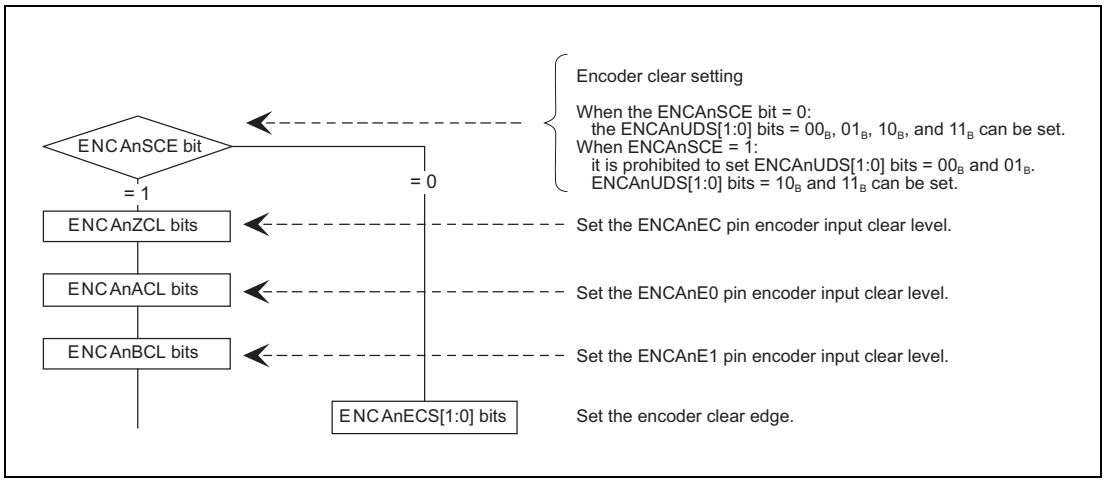



**Figure 28.8 Initial Setting Procedure for Counter Clear**

### 28.5.1.3 Setting Procedure for ENCAnCCR0 Register

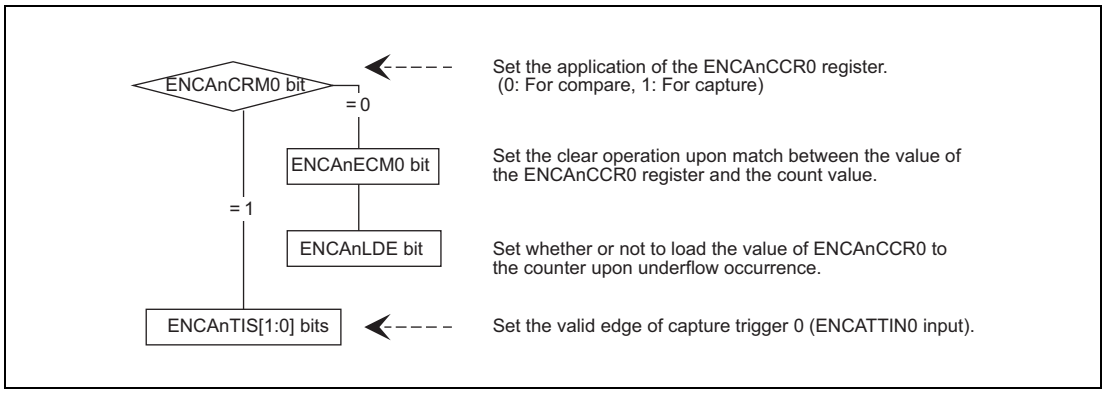



**Figure 28.9 Setting Procedure for ENCAnCCR0 Register**

### 28.5.1.4 Setting Procedure for ENCAnCCR1 Register

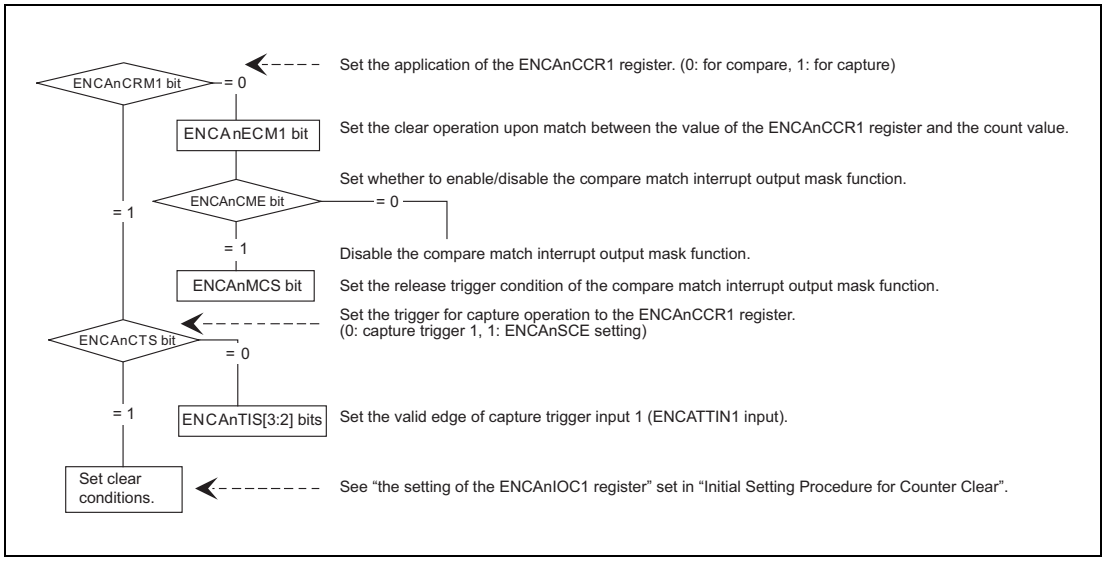



**Figure 28.10 Setting Procedure for ENCAnCCR1 Register**

# 28.6 Timing Chart

## 28.6.1 Overflow Occurrence and Overflow Flag Clear Operation

An overflow occurs when up-counting is performed when the counter value is $FFFF_H$. Once an overflow occurs, an overflow interrupt (ENCATIOV) is output and the overflow flag (ENCAnOVF) is set to 1. When the overflow clear bit (ENCAnCLOV) is set to 1, the overflow flag (ENCAnOVF) is cleared to 0.

The operations of overflow occurrence and overflow flag clearing are described below.

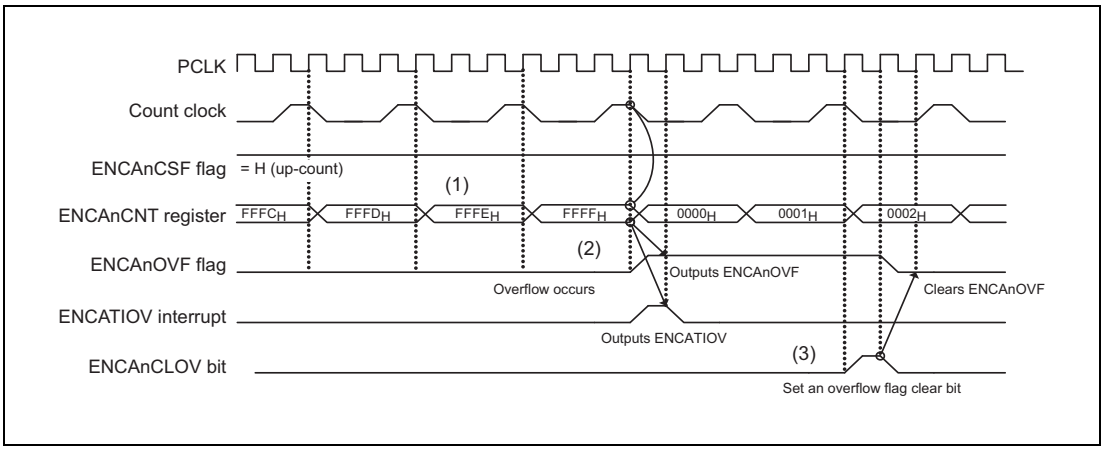



**Figure 28.11 Settings of Overflow Occurrence and Overflow Flag Clear**

(1) The count value is counted up from $FFFE_H$ to $FFFF_H$.

(2) When the count value changes from $FFFF_H$ to $0000_H$, an overflow occurs. At the same time, an overflow interrupt is output and the overflow flag is set to 1.

(3) By setting the ENCAnCLOV bit in the ENCAnFGC register to 1 by the overflow flag clearing method, the overflow flag is cleared to 0. The overflow flag is also cleared by setting the ENCAnTS bit in the ENCAnTS register to 1 when the ENCAnTE bit in the ENCAnTE register is 0, or setting the input signal of ENCATSST (simultaneous start trigger input) to “High”.

### 28.6.2 Underflow Occurrence and Underflow Flag Clear Operation

An underflow occurs when down-counting is performed when the counter value is $0000_H$.
Once an underflow occurs, an underflow interrupt (ENCATIUD) is output and the underflow flag (ENCAnUDF) is set to 1. When the underflow clear bit (ENCAnCLUD) is set to 1, the underflow flag (ENCAnUDF) is cleared to 0.

The operations of underflow occurrence and underflow flag clearing are described below.

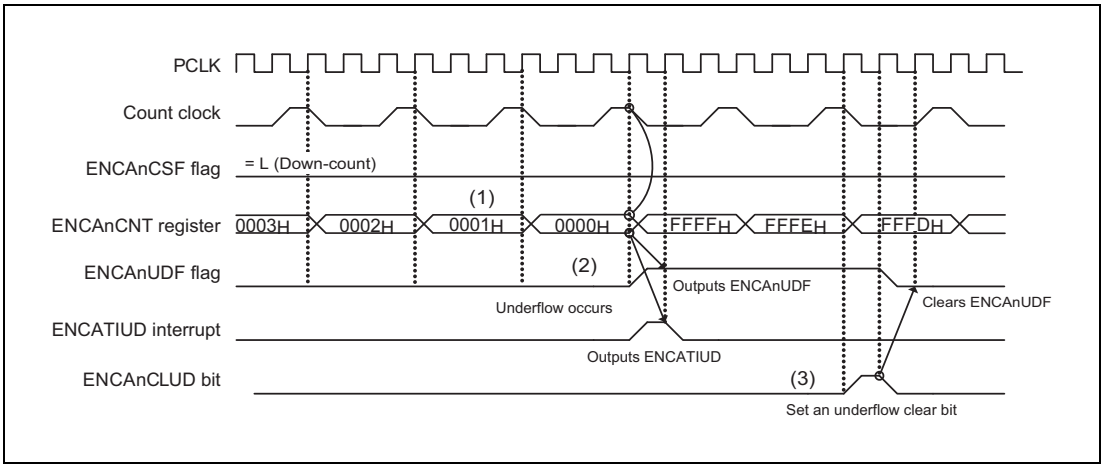



**Figure 28.12 Settings of Underflow Occurrence and Underflow Flag Clear**

(1) The count value is counted down from $0001_H$ to $0000_H$.

(2) When the count value changes from $0000_H$ to $FFFF_H$, an underflow occurs. At the same time, an underflow interrupt is output and the underflow flag is set to 1.

(3) By setting the ENCAnCLUD bit in the ENCAnFGC register to 1 by the underflow flag clearing method, the underflow flag is cleared to 0. The underflow flag is also cleared by setting the ENCAnTS bit in the ENCAnTS register to 1 when the ENCAnTE bit in the ENCAnTE register is 0, or by setting the input signal of ENCATSST (simultaneous start trigger) to “High”.

### 28.6.3 Count Clearing and Capture Operation by Encoder Clear Input (ENCAnEC Pin)

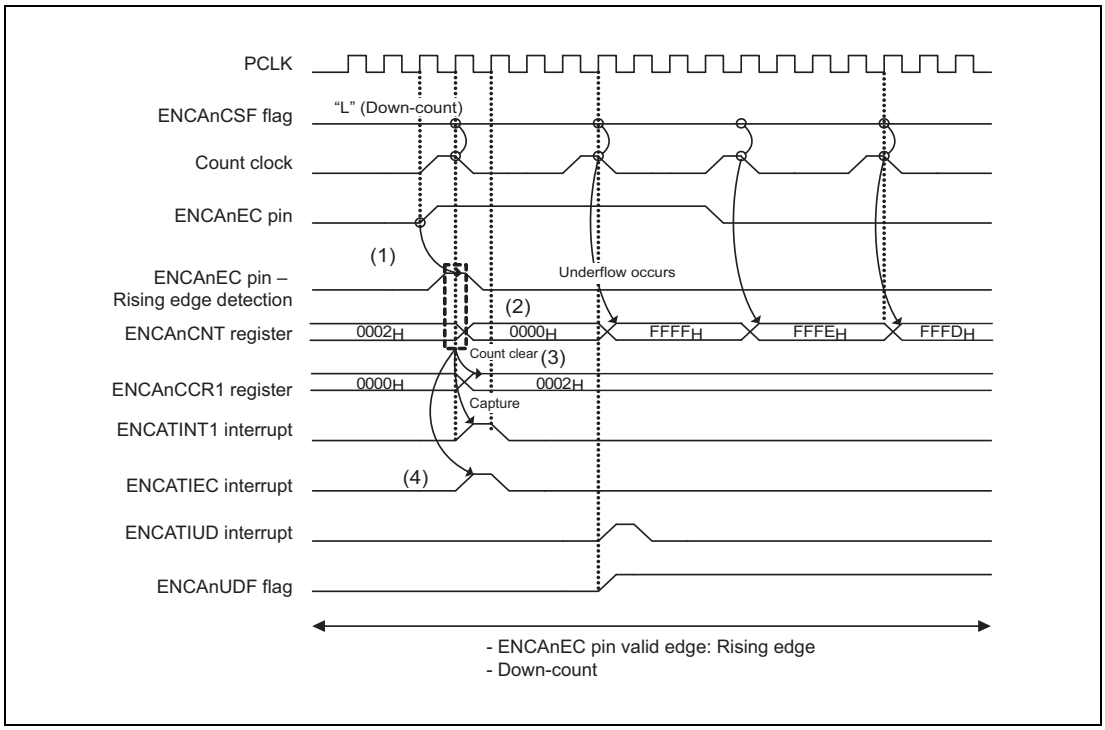



**Figure 28.13 Timing Chart of Count Clearing and Capture Operation by Encoder Clear Input (ENCAnEC Pin)**

#### Setting conditions

- ENCAnCRM1 bit in the ENCAnCTL register = 1
  (Select the ENCAnCCR1 register as capture.)
- ENCAnCTS bit in the ENCAnCTL register = 1
  (Select the ENCAnEC pin input as capture trigger input.)
- ENCAnECS1 and ENCAnECS0 bits in the ENCAnIOC1 register = $01_B$
  (Select the ENCAnEC pin input as rising edge detection.)

(1) Capture operation is performed by the rising edge of the ENCAnEC pin input trigger.

(2) Clearing is performed by the ENCAnEC pin input and the count value is set to $0000_H$.

(3) The counter value ($0002_H$) is captured in the ENCAnCCR1 register by the rising edge of the ENCAnEC pin input.

(4) At the same time, a clear interrupt (ENCATIEC) and capture interrupt (ENCATINT1) due to the ENCAnEC pin input are output.

### 28.6.4 Conflict between Overflow Occurrence and Clear Operation by Encoder Clear Input (ENCAnEC Pin)

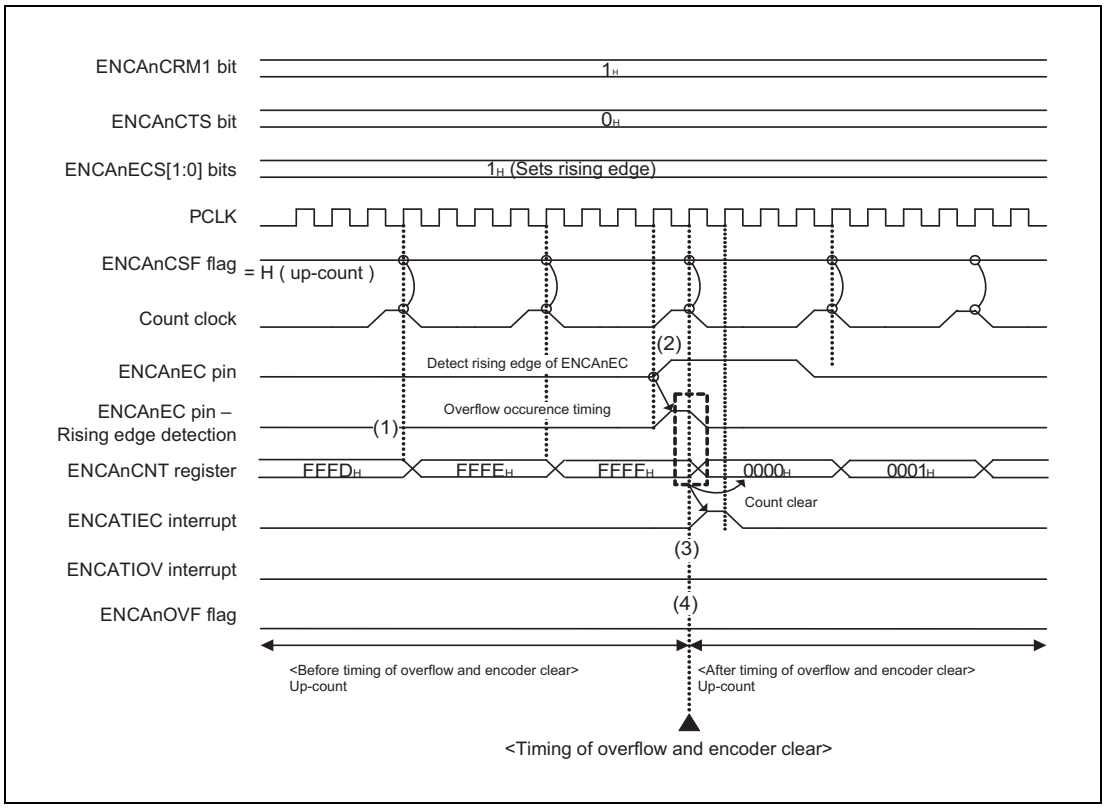



**Figure 28.14 Conflict between Overflow Occurrence and Clear Operation by Encoder Clear Input (ENCAnEC pin)**

(1) An up-count from $FFFD_H$ is continuously performed.

(2) When an overflow occurs if the count value is $FFFF_H$, and the rising edge of ENCAnEC is detected simultaneously, clear operation by the encoder clear input is performed. The counter value is cleared to $0000_H$.

(3) When the counter value is cleared by the encoder clear input, a clear interrupt (ENCATIEC) by encoder clear input is output simultaneously. Because a clear operation by the encoder clear input is performed simultaneously with the overflow occurrence, an overflow interrupt is not output (An overflow does not occur. Clear operation is performed by the encoder clear input).

(4) Because an overflow does not occur as is the case with step 3, the overflow flag is not set.

### 28.6.5 Conflict between Underflow Occurrence and Clear Operation by Encoder Clear Input (ENCAnEC Pin)

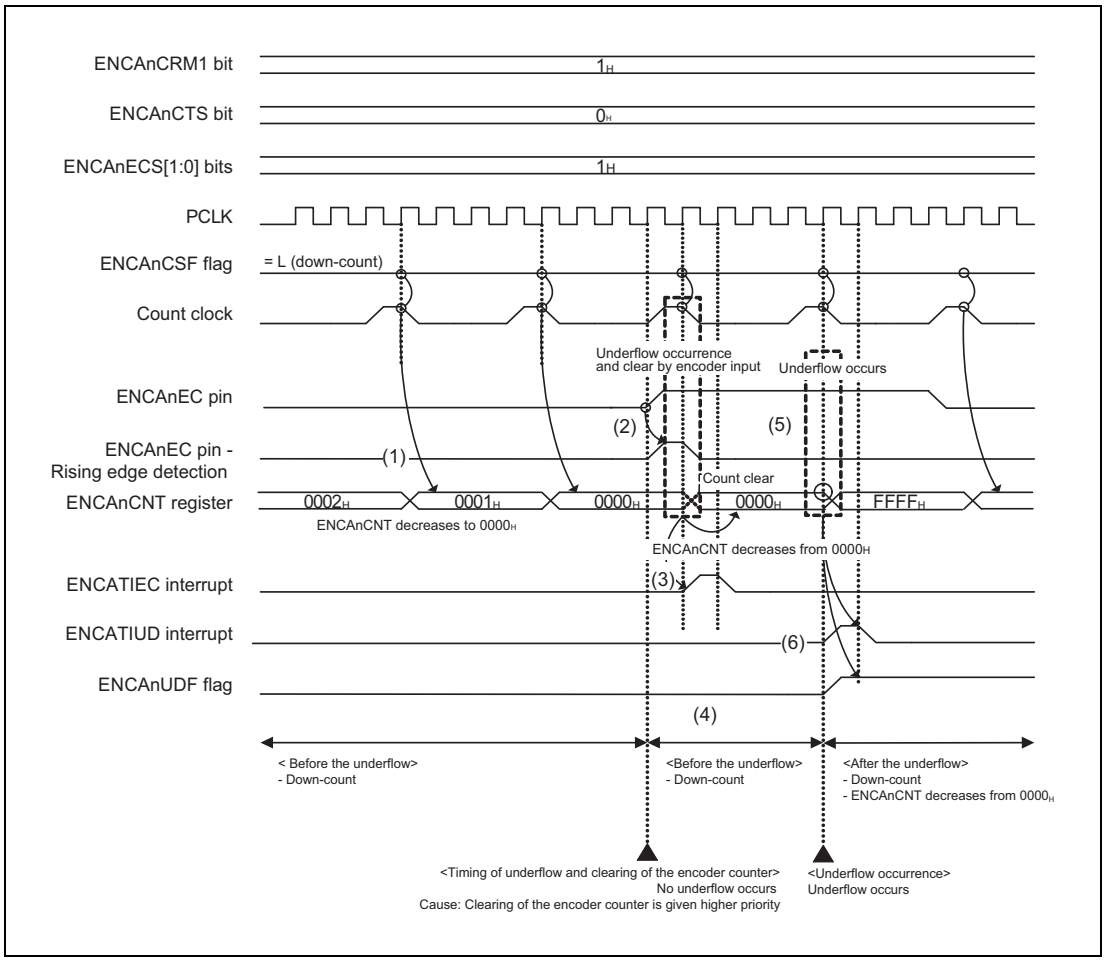



**Figure 28.15 Conflict between Underflow Occurrence and Clear Operation by Encoder Clear Input (ENCAnEC Pin)**

(1) A down-count from $0002_H$ is continuously performed.

(2) When an underflow occurs if the count value is $0000_H$, and the rising edge of ENCAnEC is detected simultaneously, clear operation by the encoder clear input is performed. Even if the next clock signal is input during clear operation, the counter value remains at $0000_H$.

(3) When the counter value is cleared by the encoder clear input, an encoder clear interrupt (ENCATIEC) is output simultaneously. Because a clear operation by the encoder clear input is performed simultaneously with the underflow occurrence, an underflow interrupt is not output (An underflow does not occur. Clear operation is performed by the encoder clear input).

(4) Because an underflow does not occur as is the case with step 3, the underflow flag is not set.

(5) When a further down-count is performed after the counter value changes to $0000_H$ by clear operation by the encoder clear input, the counter value changes from $0000_H$ to $FFFF_H$, and an underflow occurs.

(6) When an underflow occurs, an underflow interrupt (ENCATIUD) is output, and the underflow flag (ENCAnUDF) is set.

### 28.6.6 Overflow Operation Immediately after Startup

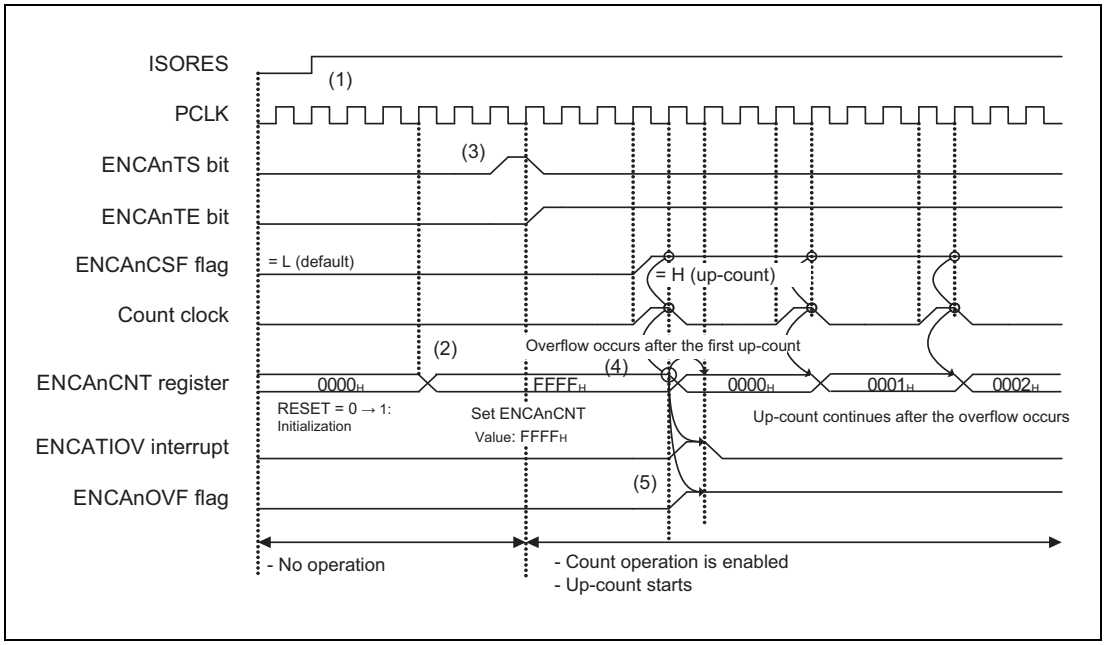



**Figure 28.16 Overflow Operation Immediately after Startup**

(1) When the ISORES value changes from “0” to “1”, the status changes from “reset asserted” to “reset deasserted”.

(2) The timer counter is set to $FFFF_H$ as the initial value.

(3) ENCAnTS is set to “1”, and operation starts. ENCAnTE changes to “1”, which indicates that operation is enabled.

(4) When an up-count is performed from $FFFF_H$ which is the initially set count value, the counter value changes from $FFFF_H$ to $0000_H$, and an overflow occurs immediately after operation starts.

(5) At the same time, by an overflow occurrence immediately after operation starts, an overflow interrupt (ENCATIOV) is output, and the overflow flag (ENCAnOVF) is set.

### 28.6.7 Underflow Operation Immediately after Startup

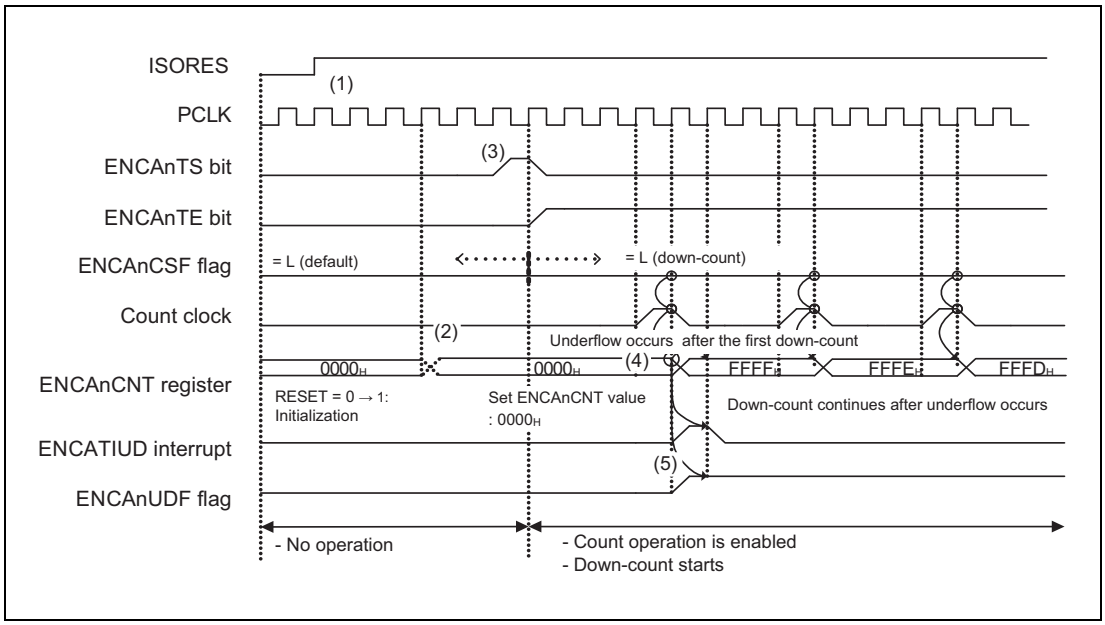



**Figure 28.17 Underflow Operation Immediately after Startup**

(1) When the ISORES value changes from “0” to “1”, the status changes from “reset asserted” to “reset deasserted”.

(2) The timer counter is set to $0000_H$ as the initial value.

(3) ENCAnTS is set to “1”, and operation starts. ENCAnTE changes to “1”, which indicates that operation is enabled.

(4) When a down-count is performed from $0000_H$ which is the initially set count value, the counter value changes from $0000_H$ to $FFFF_H$, and an underflow occurs immediately after operation starts.

(5) At the same time, by an underflow occurrence immediately after operation starts, an underflow interrupt (ENCATIUD) is output, and the underflow flag (ENCAnUDF) is set.

### 28.6.8 Using the ENCAnLDE Function Immediately after Startup

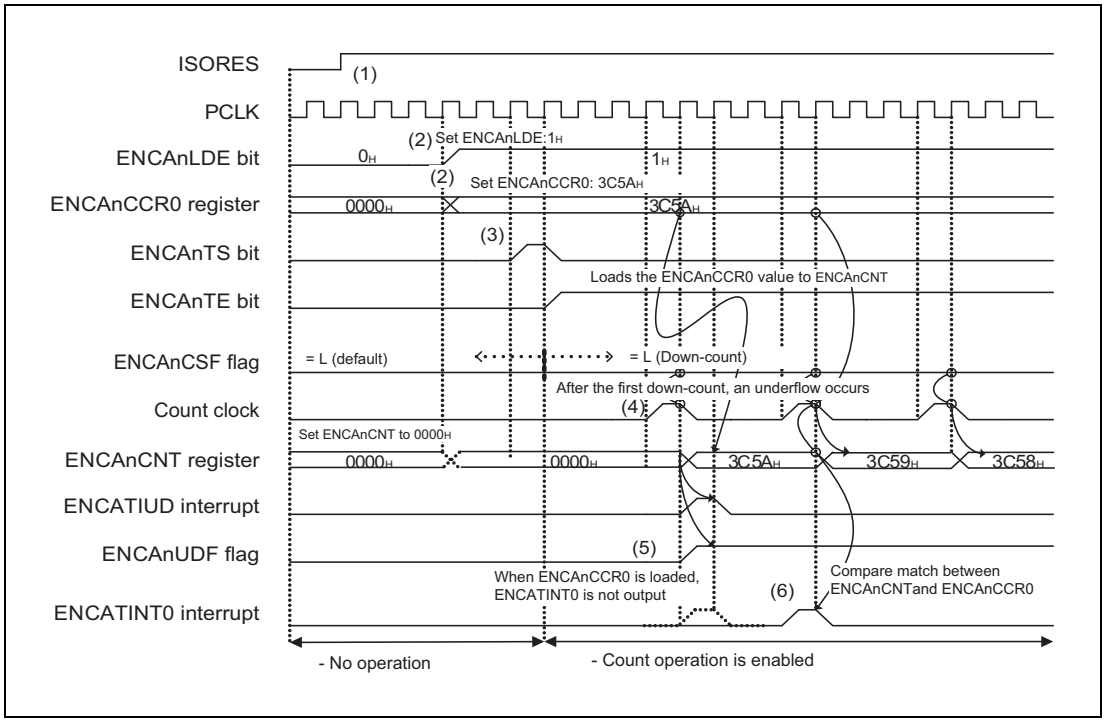



**Figure 28.18 Using the ENCAnLDE Function Immediately after Startup**

(1) When the ISORES value changes from “0” to “1”, the status changes from “reset asserted” to “reset deasserted”.

(2) The load enable bit (ENCAnLDE) is set to “1”, capture/compare register 0 (ENCAnCCR0) is set to $3C5A_H$, and the timer counter is set to the initial value $0000_H$.

(3) ENCAnTS is set to “1”, and operation starts. ENCAnTE changes to “1”, which indicates that operation is enabled.

(4) When a down-count is performed from $0000_H$ which is the initially set count value, an underflow occurs immediately after operation starts. Because ENCAnLDE is set to “1”, the ENCAnCCR0 value, $3C5A_H$, is loaded to the timer counter (ENCATINT0 is not output during loading).

(5) At the same time, by an underflow occurrence immediately after operation starts, an underflow interrupt (ENCATIUD) is output, and the underflow flag (ENCAnUDF) is set (after an underflow occurs, down-count operation from the loaded value ($3C5A_H$) continues).

(6) After the ENCAnCCR0 value is loaded to ENCAnCNT, a match with ENCAnCCR0 is detected, and ENCATINT0 is output.

## 28.6.9 ENCAnLDE Function (Loading Count Value)

### (1) <When ENCAnLDE = 0>

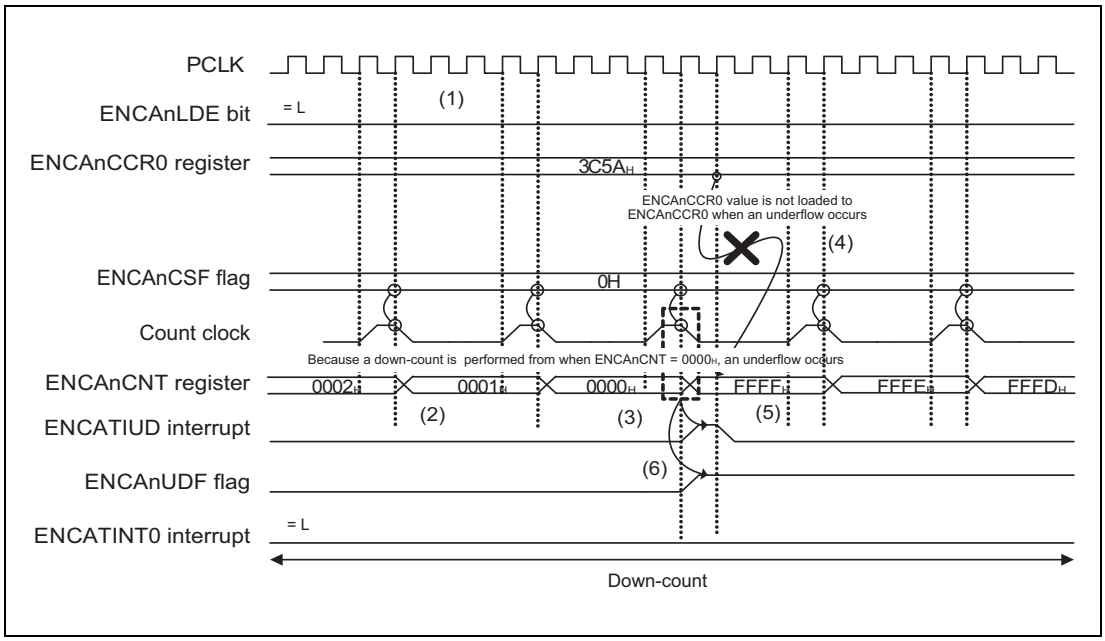



**Figure 28.19 ENCAnLDE Function (when ENCAnLDE = 0)**

(1) ENCAnLDE is set to “0” (even if an underflow occurs, the ENCAnCCR0 value is not loaded).

(2) A down-count is performed: $0002_H \rightarrow 0001_H \rightarrow 0000_H$

(3) When a further down-count is performed after the counter value changes to $0000_H$, an underflow occurs.

(4) Because ENCAnLDE is set to “0”, the setting value of the ENCAnCCR0 register is not loaded to the counter even if an underflow occurs.

(5) Operation changes to underflow operation (counter value: $0000_H \rightarrow FFFF_H$).

(6) An underflow interrupt (ENCATIUD) is output, and the underflow flag (ENCAnUDF) is set.

**(2) <When ENCAnLDE = 1>**

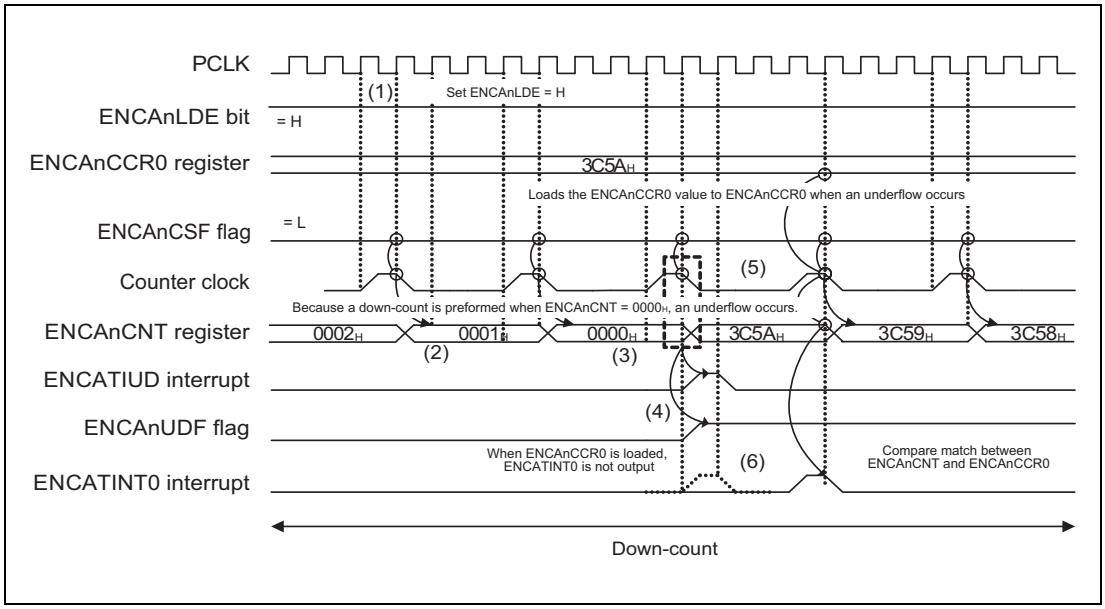



**Figure 28.20 ENCAnLDE Function (when ENCAnLDE = 1)**

(1) ENCAnLDE is set to “1” (if an underflow occurs, the ENCAnCCR0 value is loaded to the counter).

(2) A down-count is performed: $0002_H \rightarrow 0001_H \rightarrow 0000_H$

(3) When a further down-count is performed after the counter value changes to $0000_H$, an underflow occurs.

(4) An underflow interrupt is output, and the underflow flag is set.

(5) Because ENCAnLDE is set to “1”, the setting value of the ENCAnCCR0 register is loaded to the counter if an underflow occurs. ENCAnCNT is set to $3C5A_H$.

(6) After the ENCAnCCR0 value is set to ENCAnCNT, if the ENCAnCNT value matches the ENCAnCCR0 value on a count clock, a compare match interrupt (ENCATINT0) is output.

### 28.6.10 Conflict between ENCAnLDE Function (Loading Counter Value) and Rewrite of ENCAnCCR0 Register

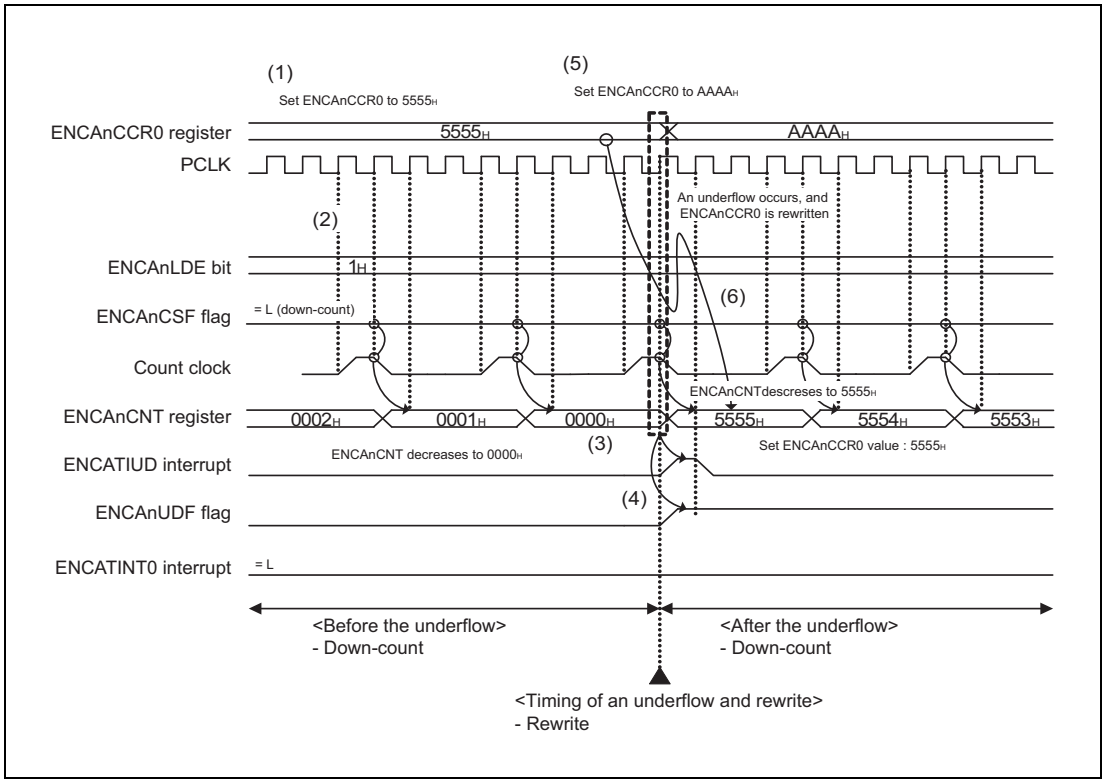



**Figure 28.21 Conflict between ENCAnLDE Function and Rewrite of ENCAnCCR0 Register**

(1) The ENCAnCCR0 register is currently set to $5555_H$.

(2) ENCAnLDE is currently set to “1”.

(3) A down-count is performed ($0002_H \rightarrow 0001_H \rightarrow 0000_H$), and an underflow occurs.

(4) An underflow interrupt (ENCATIUD) is output, and the underflow flag (ENCAnUDF) is set.

(5) When an underflow occurs, the ENCAnCCR0 register value is changed from $5555_H$ to $AAAA_H$.

(6) Additionally, when an underflow occurs, the ENCAnCCR0 value before the rewrite was performed ($5555_H$) is set in ENCAnCNT.

### 28.6.11 Conflict between ENCAnLDE Function (Loading Counter Value) and Clear Operation by Encoder Clear Input (ENCAnEC Pin)

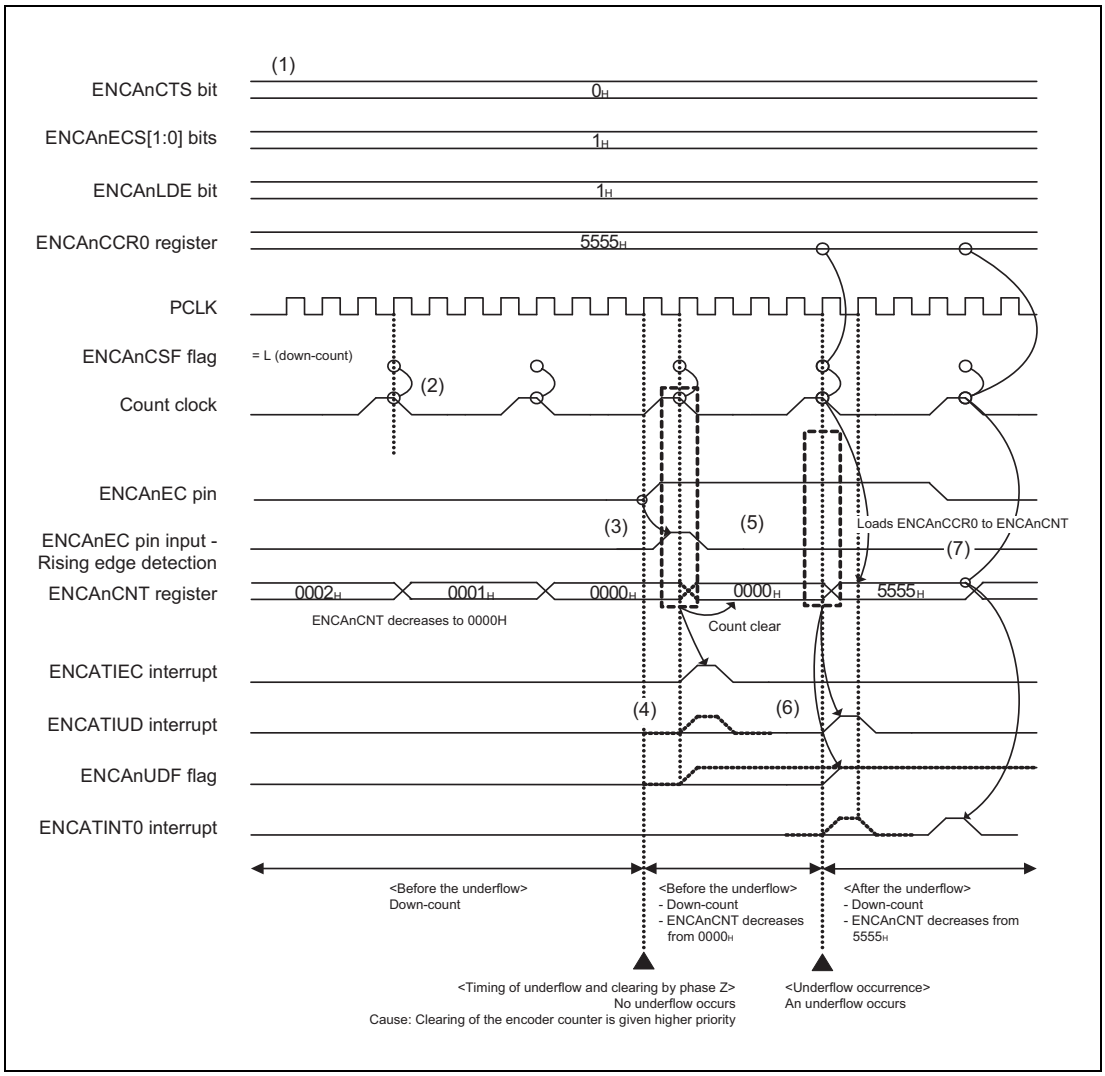



**Figure 28.22 Conflict between ENCAnLDE Function and Clear Operation by Encoder Clear Input**

(1) The values are set as follows: ENCAnCTS = 0, ENCAnECS[1:0] = $01_B$, ENCAnLDE = 1, and ENCAnCCR0 = $5555_H$.

(2) A down-count is performed: $0002_H \rightarrow 0001_H \rightarrow 0000_H$

(3) When the count value becomes $0000_H$, the rising edge of ENCAnEC pin is detected, and clear operation by the encoder clear input is performed.

(4) Because a count clear is performed when the count value reaches $0000_H$, a counter clear interrupt (ENCATIEC) by the encoder clear input is output. An underflow does not occur because a down-count is not performed when the count value is $0000_H$. Therefore, an underflow interrupt (ENCATIUD) is not output, and the underflow flag (ENCAnUDF) is not set.

(5) After the count value is cleared to $0000_H$ by clear operation by the encoder clear input, a down-count is performed and an underflow occurs.

(6) An underflow interrupt (ENCATIUD) is output, and the underflow flag (ENCAnUDF) is set.

(7) Because ENCAnLDE = “1”, if an underflow occurs, the ENCAnCCR0 value is loaded to ENCAnCNT.

(8) After the ENCAnCCR0 value is set to ENCAnCNT, a compare match is detected according to the count clock. If the ENCAnCNT value matches the ENCAnCCR0 value, a compare match interrupt (ENCATINT0) is output.

### 28.6.12 Up-count after Conflict between ENCAnLDE Function (Loading Counter Value) and Clear Operation by Encoder Clear Input

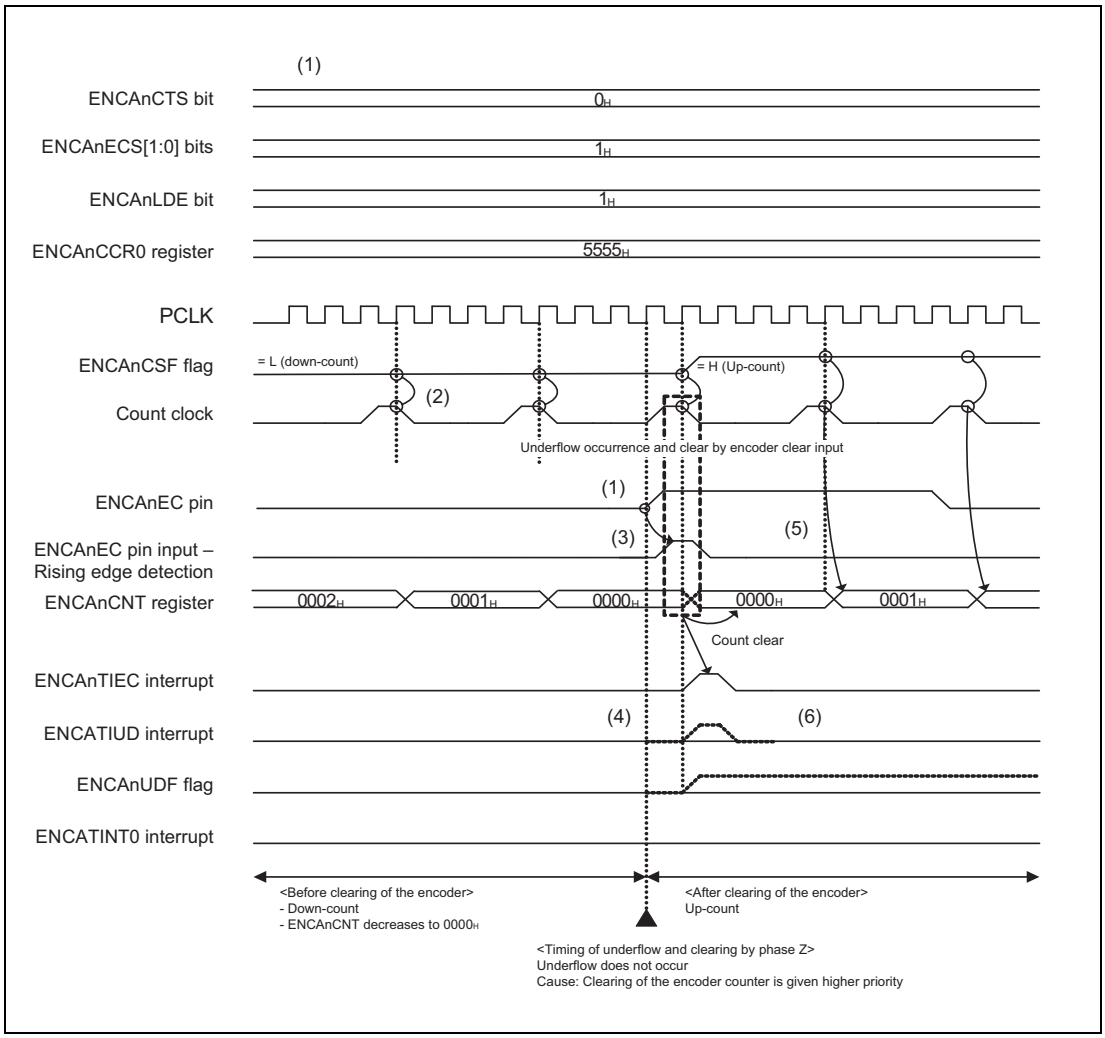



**Figure 28.23 Up-count after Conflict between ENCAnLDE Function and Encoder Clear**

(1) The values are set as follows: ENCAnCTS = 0, ENCAnECS[1:0] = $01_B$, ENCAnLDE = 1, and ENCAnCCR0 = $5555_H$.

(2) A down-count is performed: $0002_H \rightarrow 0001_H \rightarrow 0000_H$

(3) When the count value becomes $0000_H$, the rising edge of ENCAnEC pin is detected, and clear operation by the encoder clear input is performed.

(4) Because a count clear is performed when the count value reaches $0000_H$, a counter clear interrupt (ENCATIEC) by the encoder clear input is output. An underflow does not occur because a down-count is not performed when the count value is $0000_H$. Therefore, an underflow interrupt (ENCATIUD) is not output, and the underflow flag (ENCAnUDF) is not set.

(5) After the count value is cleared to $0000_H$ by clear operation by the encoder clear input, an up-count is performed.

(6) An underflow interrupt (ENCATIUD) is not output, and the underflow flag (ENCAnUDF) is not set.

### 28.6.13 Capture Operation between Count Clocks (ENCAnCCR1)

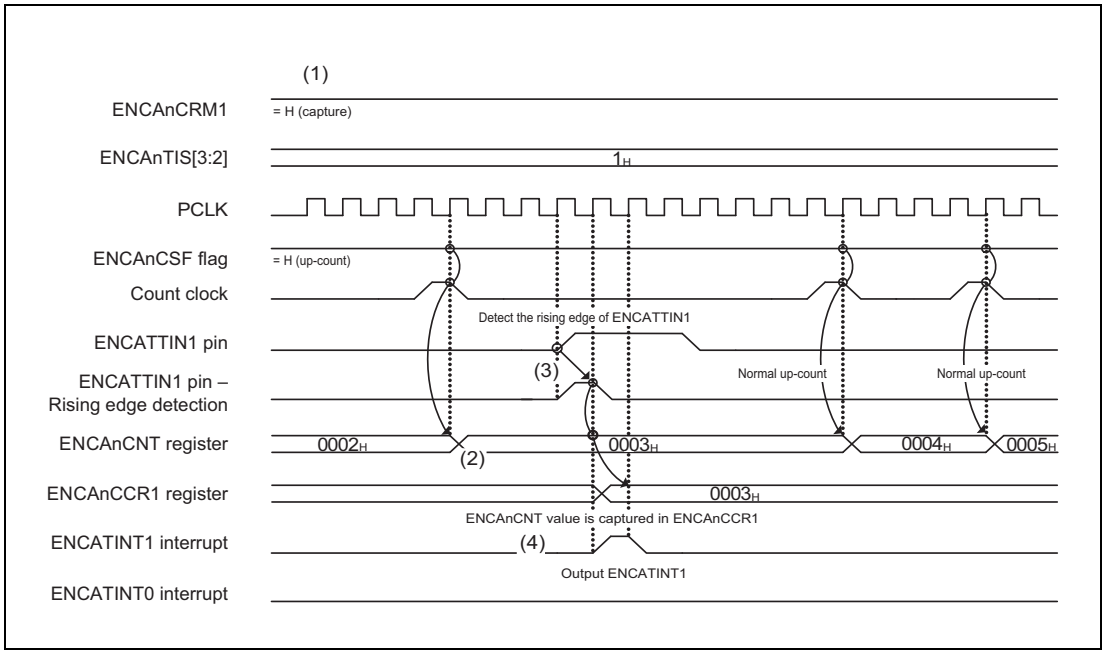



**Figure 28.24 Capture Operation between Count Clocks (ENCAnCCR1)**

(1) The values are set as follows: ENCAnCRM1 = 1, and ENCAnTIS[3:2] = $01_B$.

(2) An up-count is performed.

(3) The rising edge of the ENCATTIN1 input is detected, and the count value is captured in ENCAnCCR1.

(4) An interrupt (ENCATINT1) corresponding to the capture to the ENCAnCCR1 register is output.

### 28.6.14 Capture Operation between Count Clocks (ENCAnCCR0)

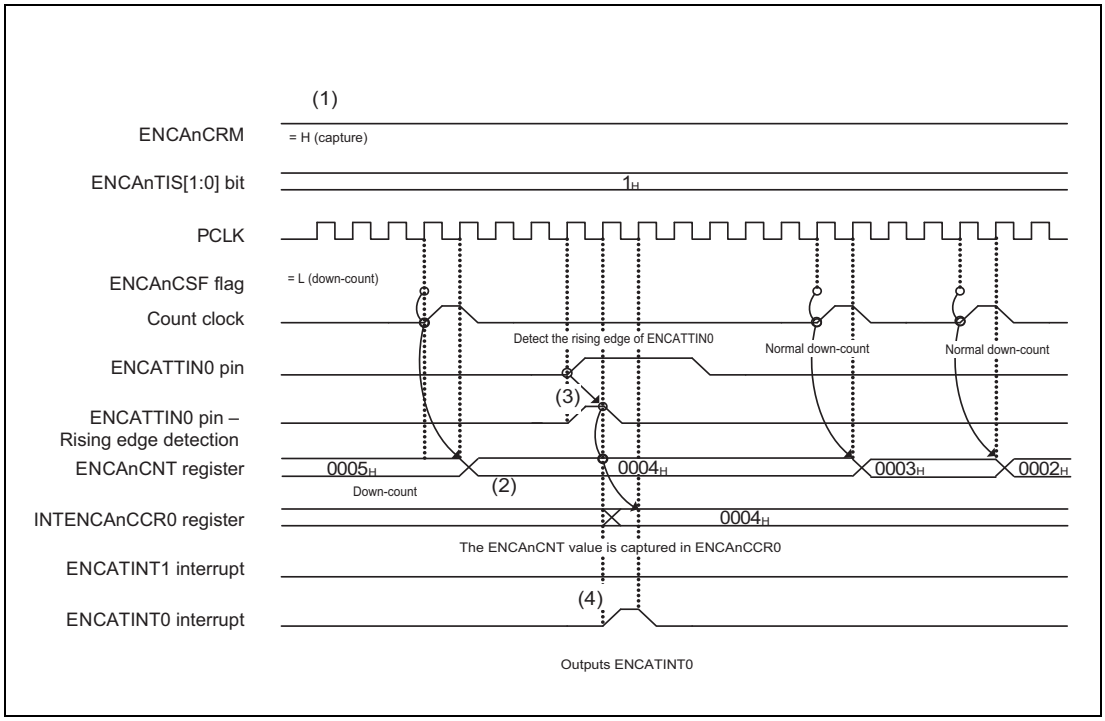



**Figure 28.25 Capture Operation between Count Clocks (ENCAnCCR0)**

(1) The values are set as follows: ENCAnCRM0 = 1, and ENCAnTIS[1:0] = $01_B$.

(2) A down-count is performed.

(3) The rising edge of the ENCATTIN0 input is detected, and the count value is captured in ENCAnCCR0.

(4) An interrupt (ENCATINT0) corresponding to the capture to the ENCAnCCR0 register is output.

### 28.6.15 Encoder Operation when Compare Match Clear Control is Enabled and ENCAnCTS = 0

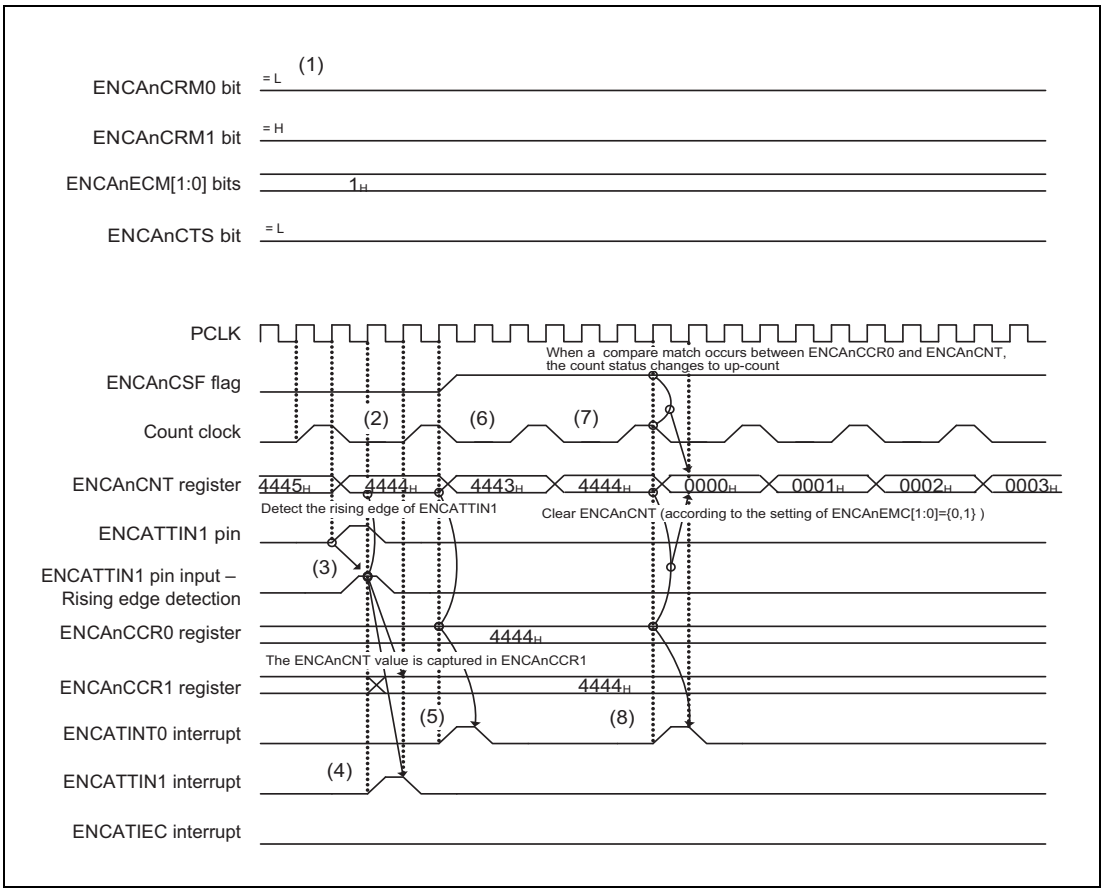



**Figure 28.26 Encoder Operation when Compare Match Clear Control is Enabled and ENCAnCTS = 0**

(1) The values are set as follows: ENCAnCCR0 = $4444_H$, ENCAnCRM0 = 0, ENCAnCRM1 = 1, ENCAnECM[1:0] = $01_B$, and ENCAnCTS = 0.

(2) A down-count is performed.

(3) The rising edge of the ENCATTIN1 input is detected, and the ENCAnCNT value ($4444_H$) is captured in the ENCAnCCR1 register.

(4) An interrupt signal (ENCATINT1) corresponding to the capture to the ENCAnCCR1 register is output.

(5) When a compare match occurs between ENCAnCNT (counted down from $4445_H$ to $4444_H$) and ENCAnCCR0 ($4444_H$), a compare match interrupt (ENCATINT0) with ENCAnCCR0 is output.

(6) The count operation changes to up-count.

(7) When ENCAnCNT is counted up from $4443_H$ to $4444_H$, a compare match with ENCAnCCR0 occurs again. Because the count operation is up-count when the compare match occurs, the count value is cleared according to the setting of ENCAnECM1 and ENCAnECM0 ($01_B$), and the ENCAnCNT value changes to $0000_H$.

(8) When ENCAnCNT changes to $4444_H$, a compare match interrupt (ENCATINT0) with ENCAnCCR0 is output.

### 28.6.16 Encoder Operation when Compare Match Clear Control is Enabled and ENCAnCTS = 1

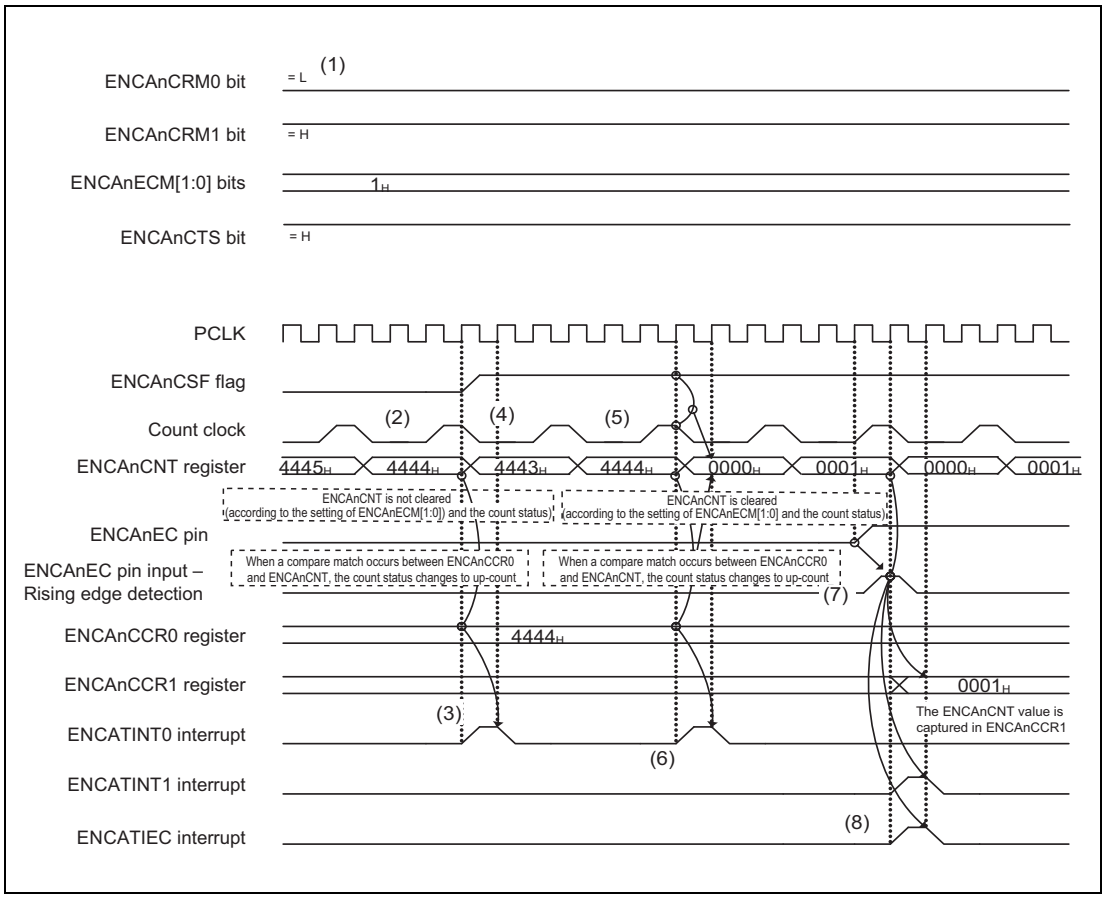



**Figure 28.27 Encoder Operation when Compare Match Clear Control is Enabled and ENCAnCTS = 1**

(1) The values are set as follows: ENCAnCCR0 = $4444_H$, ENCAnCRM0 = 0, ENCAnCRM1 = 1, ENCAnECM[1:0] = $01_B$, and ENCAnCTS = 1.

(2) A down-count is performed.

(3) When a compare match occurs between ENCAnCNT (counted down from $4445_H$ to $4444_H$) and ENCAnCCR0 ($4444_H$), a compare/capture interrupt (ENCATINT0) is output.

(4) The count operation changes to up-count.

(5) When ENCAnCNT is counted up from $4443_H$ to $4444_H$, a compare match with ENCAnCCR0 occurs again. Because the count operation is up-count when the compare match occurs, the count value is cleared according to the setting of ENCAnECM1 and ENCAnECM0 ($01_B$), and the ENCAnCNT value changes to $0000_H$.

(6) When ENCAnCNT changes to $4444_H$, a compare match interrupt (ENCATINT0) with ENCAnCCR0 is output.

(7) After the count value is cleared, an up-count is performed, and the count value changes to $0001_H$. At this point, the ENCAnCNT value ($0001_H$) is captured in ENCAnCCR1 by detecting the rising edge of the ENCAnEC signal, and the counter is cleared to $0000_H$.

(8) An interrupt (ENCATINT1) corresponding to the capture to the ENCAnCCR1 register and a clear interrupt (ENCATIEC) by ENCAnEC are output.

### 28.6.17 Encoder Operation when Compare Match Clear Control is Disabled

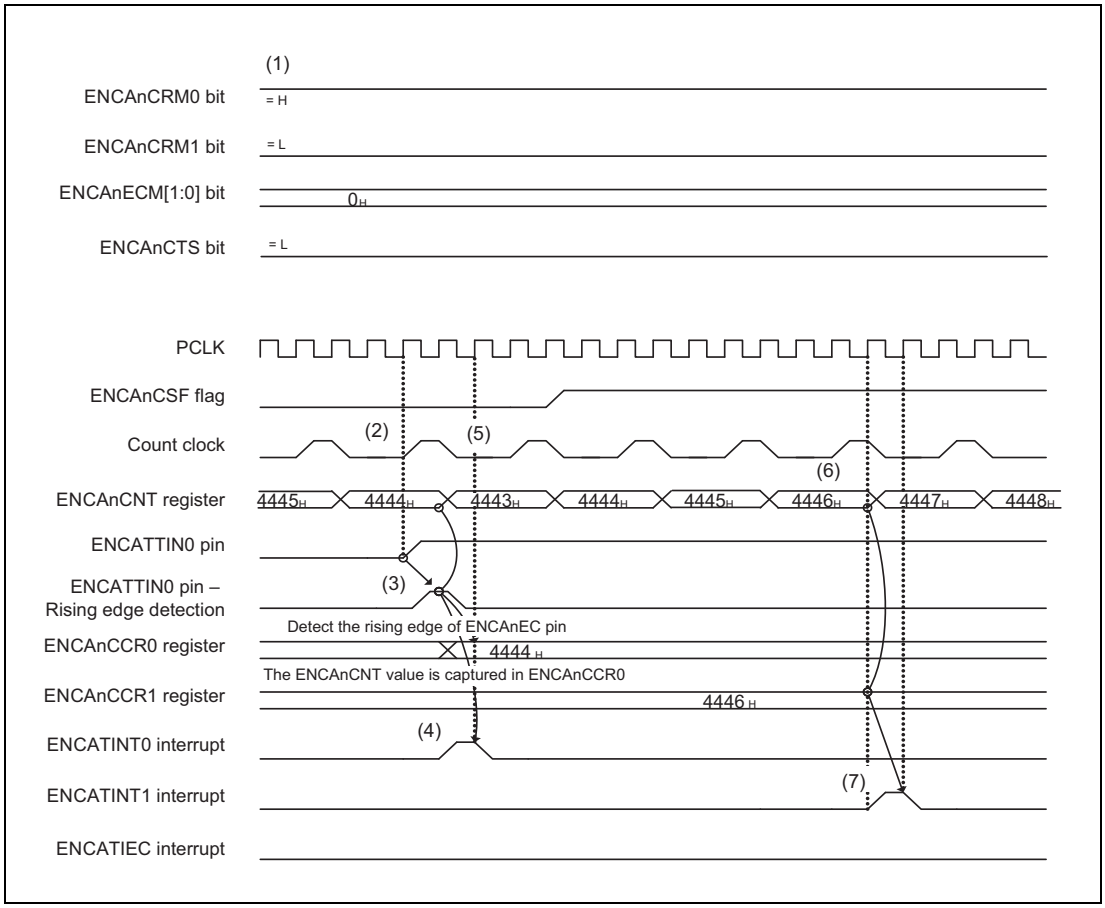



**Figure 28.28 Encoder Operation when Compare Match Clear Control is Disabled**

(1) The values are set as follows: ENCAnCCR1 = $4446_H$, ENCAnCRM0 = 1, ENCAnCRM1 = 0, ENCAnECM[1:0] = $00_B$, and ENCAnCTS = 0.

(2) A down-count is performed.

(3) When the rising edge of ENCATTIN0 is detected, the ENCAnCNT value ($4444_H$) is captured in ENCAnCCR0.

(4) An interrupt signal (ENCATINT0) corresponding to the capture to the ENCAnCCR0 register is output.

(5) The count operation changes to up-count.

(6) When ENCAnCNT changes to $4446_H$, a compare match with ENCAnCCR1 is detected.

(7) A compare match interrupt (ENCATINT1) with ENCAnCCR1 is output.

## 28.6.18 Capture Operation Performed upon Clearing by ENCAnEC, ENCAnE0, ENCAnE1 when ENCAnSCE = 1

### 28.6.18.1 Accompanying capture operation

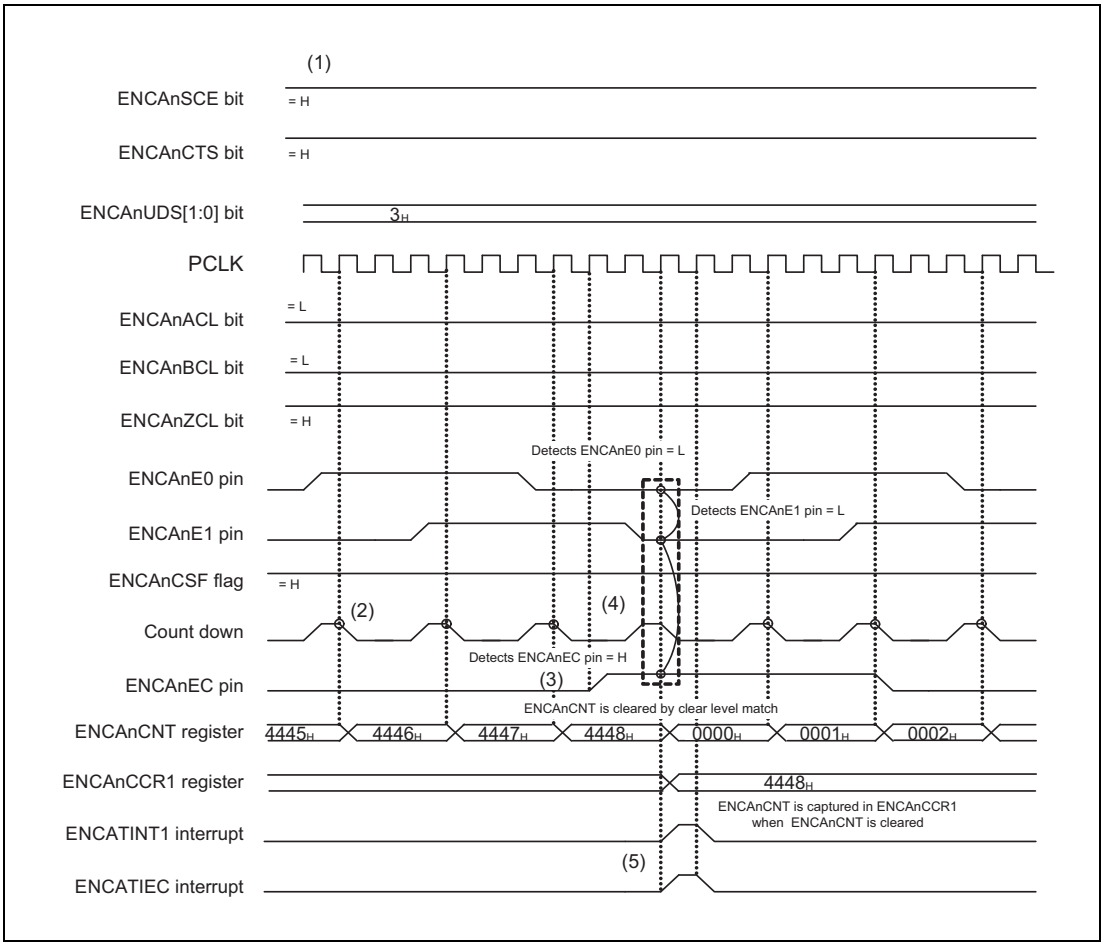



**Figure 28.29 Capture Operation Performed upon Clearing by ENCAnEC, ENCAnE0, ENCAnE1 when ENCAnSCE = 1**

(1) The values are set as follows: ENCAnSCE = 1, ENCAnCTS = 1, ENCAnUDS[1:0] =$11_B$, ENCAnACL = 0, ENCAnBCL = 0, and ENCAnZCL = 1.

(2) An up-count is performed.

(3) The count value is not cleared upon the rising edge of ENCAnEC.

(4) When ENCAnE0, ENCAnE1 and ENCAnEC reach the set clear level, the count value is cleared. The count value is captured in ENCAnCCR1 at the time of the clearing.

(5) At the time of the clearing, an interrupt (ENCATINT1) corresponding to the capture to the ENCAnCCR1 register and a clear interrupt (ENCATIEC) by ENCAnEC are output.

### 28.6.18.2 When the Timing of the ENCAnEC Input is Later than that of the ENCAnE1 Input during Up-count (When ENCAnACL = 1, ENCAnBCL = 0, ENCAnZCL = 1, and ENCAnUDS[1:0] = $11_B$)

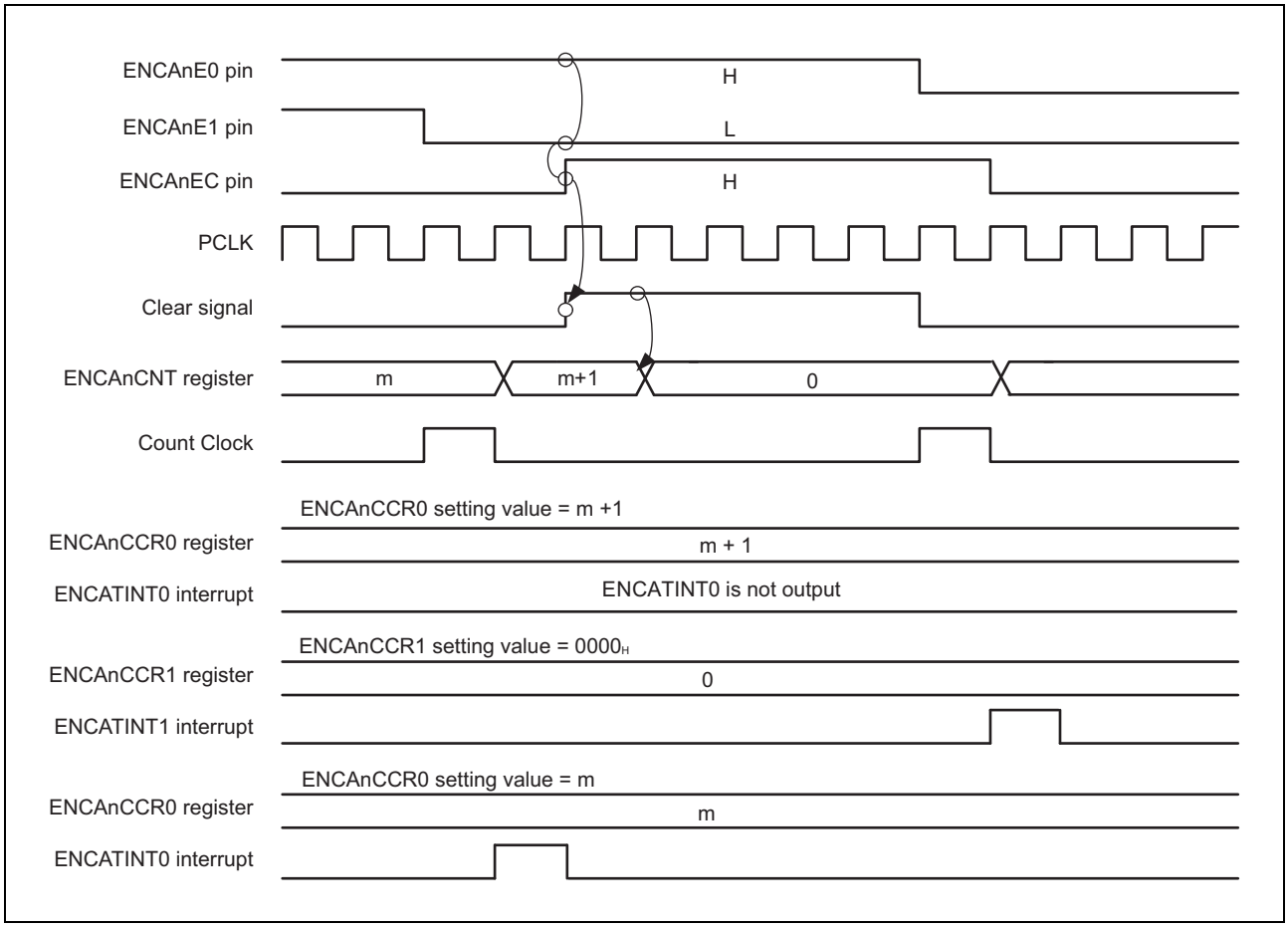



**Figure 28.30 Clearing Timing for when the Timing of the ENCAnEC Input is Later than that of the ENCAnE1 Input during Up-count**

### 28.6.18.3 When the Timing of the ENCAnEC Input is the Same as that of the ENCAnE1 Input during Up-count (When ENCAnACL = 1, ENCAnBCL = 0, ENCAnZCL = 1, and ENCAnUDS[1:0] = $11_B$)

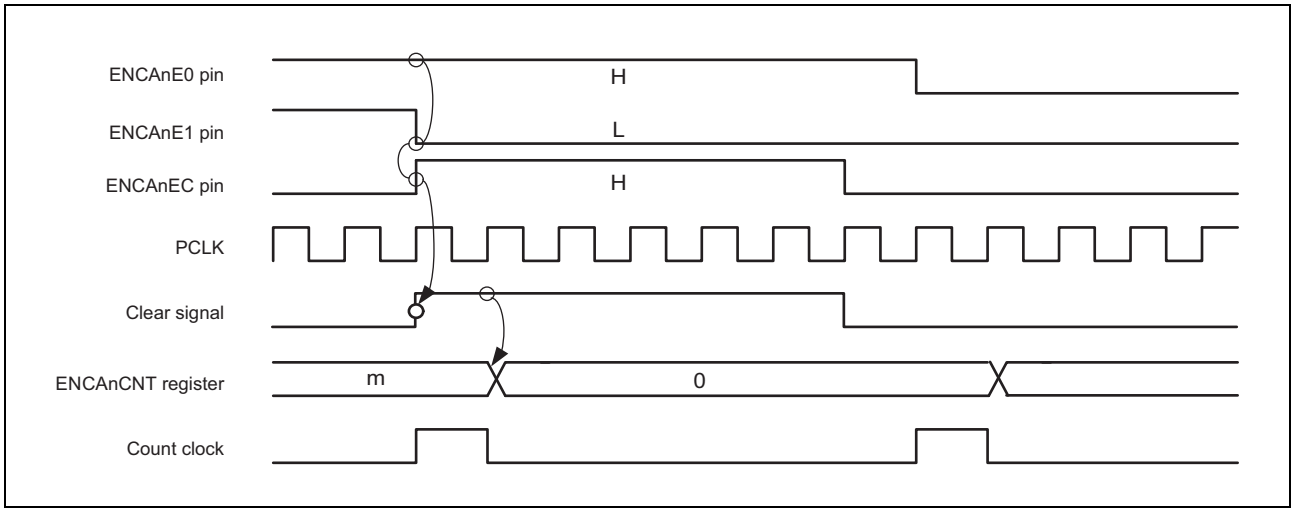



Figure 28.31 Clearing Timing for when the Timing of the ENCAnEC Input is the Same as that of the ENCAnE1 Input during Up-count

### 28.6.18.4 When the Timing of the ENCAnEC Input is Earlier than that of the ENCAnE1 Input during Up-count (When ENCAnACL = 1, ENCAnBCL = 0, ENCAnZCL = 1, and ENCAnUDS[1:0] = $11_B$)

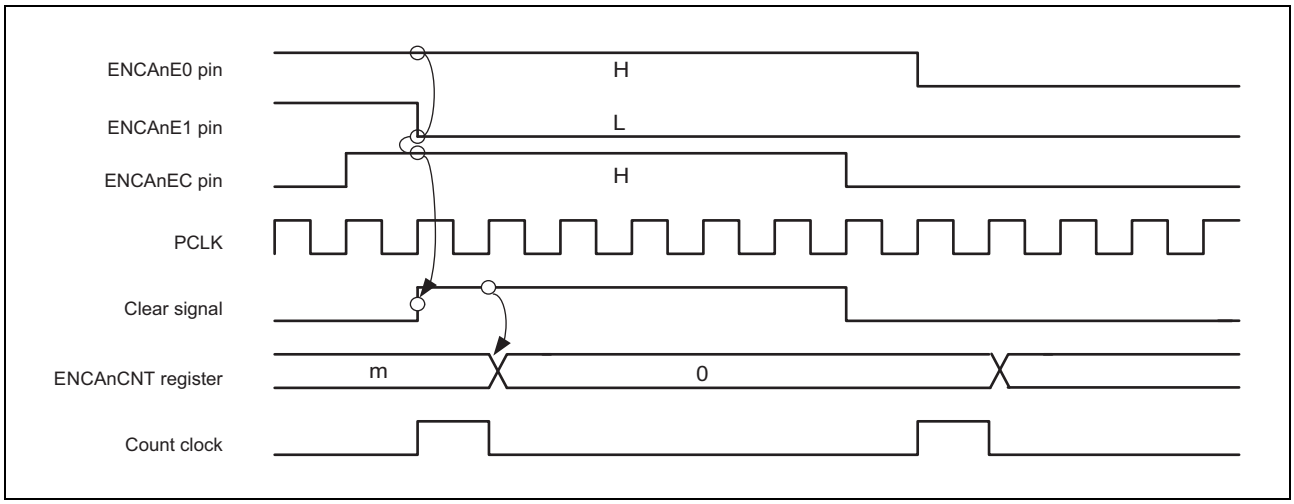



Figure 28.32 Clearing Timing for when the Timing of the ENCAnEC Input is Earlier than that of the ENCAnE1 Input during Up-count

### 28.6.18.5 When the Timing of the ENCAnEC Input is Later than that of the ENCAnE1 Input during Down-count (When ENCAnACL = 1, ENCAnBCL = 0, ENCAnZCL = 1, and ENCAnUDS[1:0] = $11_B$)

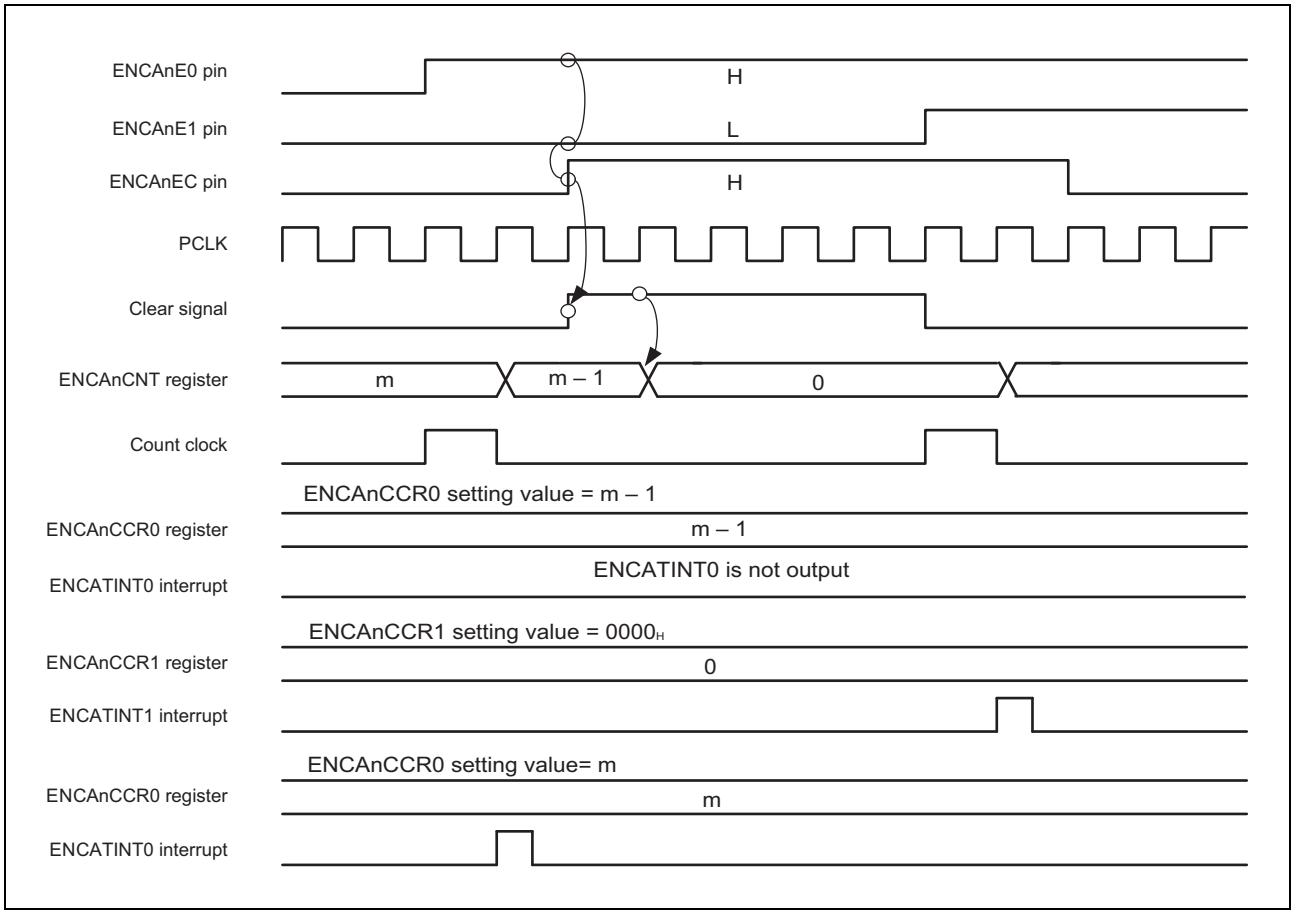



**Figure 28.33 Clearing Timing for when the Timing of the ENCAnEC Input is Later than that of the ENCAnE1 Input during Down-count**

### 28.6.19 Capture Operation Performed upon Clearing by ENCAnEC when ENCAnSCE = 0

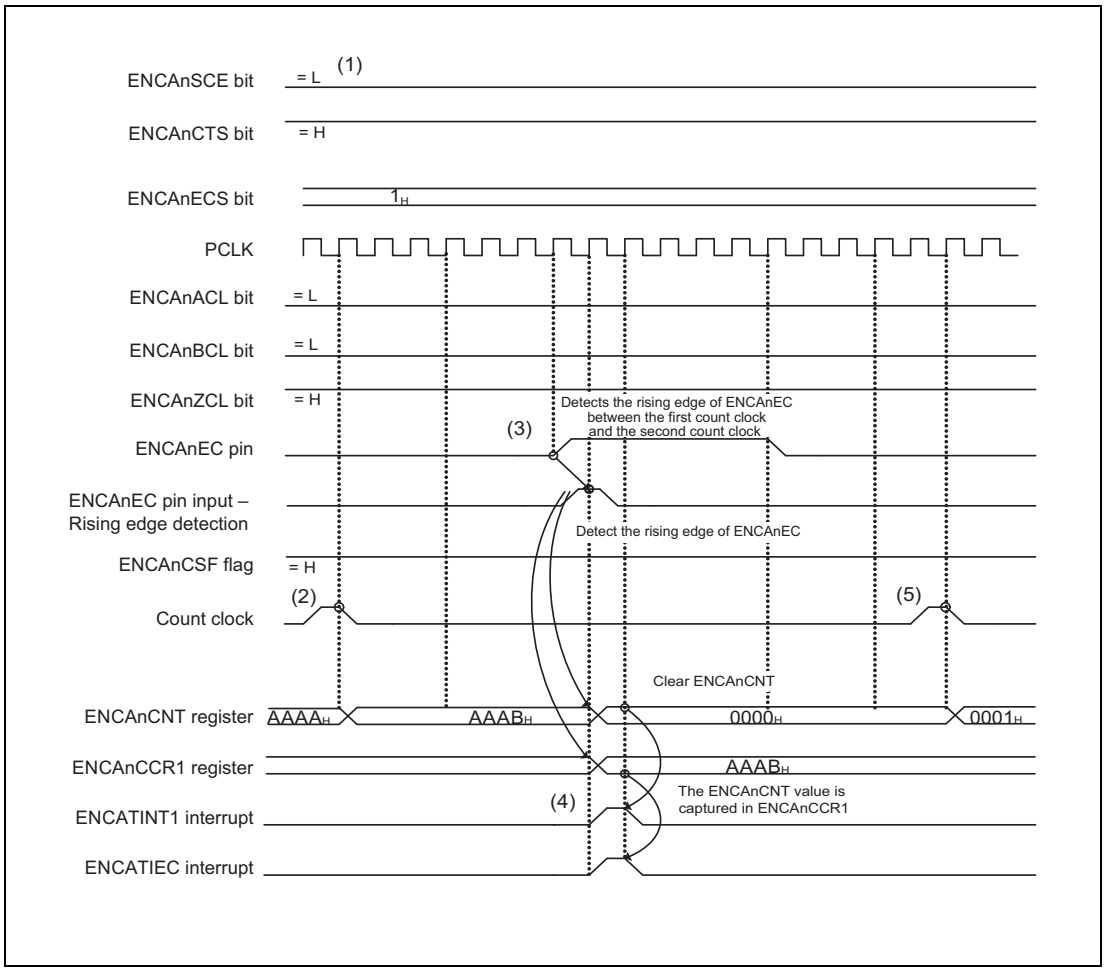



**Figure 28.34 Capture Operation Performed upon Clearing by ENCAnEC when ENCAnSCE = 0**

(1) The values are set as follows: ENCAnSCE = 0, ENCAnCTS = 1, and ENCAnECS[1:0] = $01_B$.

(2) An up-count is performed.

(3) The rising edge of the ENCAnEC input is detected, and the ENCAnCNT value ($AAAB_H$) is captured in the ENCAnCCR1 register. Concurrently, clear operation by ENCAnEC is performed, and ENCAnCNT is cleared to $0000_H$.

(4) A capture interrupt 1 (ENCATINT1) to the ENCAnCCR1 register and an encoder clear interrupt (ENCATIEC) by ENCAnEC are output.

(5) After the count value is cleared, an up-count is performed, and the count value changes to $0001_H$.

# Section 29 Motor Control

This section contains a generic description of the Motor Control.

The first part in this section describes all RH850/F1K specific properties, such as the number of units, register base addresses, etc.

The remainder of the section describes the functions and registers of the Motor Control.

## 29.1 Features of RH850/F1K Motor Control

### 29.1.1 Number of Units and Channels

Motor control function comprises the timer motor control units (TAPA) and the peripheral interconnection unit (PIC) to connect the TAPA unit to peripheral timers, and generates motor control waveforms by using a combination of peripheral timers and A/D converters.

This microcontroller has the following number of units of the TAPA and PIC units.

**Table 29.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| **TAPA** | | | |
| Number of Units | 1 | | |
| Name | TAPAn (n = 0) | | |
| **PIC** | | | |
| Number of Units | 1 | | |
| Name | PIC0 | | |

**Table 29.2 Indices**

| Index | Description |
|---|---|
| n | Throughout this section, the unit of a timer and A/D converter used by TAPA and the motor control function is identified by the index “n” (n = 0): for example, TAPAnCTL0 is TAPAn control register 0. |
| m | The channel of a used timer and A/D converter is identified by the index “m”. For example, the TAUDn channel is described as CHm. |
| x | The scan group of an A/D converter is identified by the index “x” (x = 1 to 3). |
| j | The scan trigger number of an A/D converter is identified by the index “j” (j = 0 to 2). |

The following table lists the values indicated by the indices of each product.

**Table 29.3 Indices of Products**

| Indices of Each Product |
|---|
| **All Products** |
| m = 0 to 15<br>(e.g. TAUDn) |
| x = 1 to 3 |
| j = 0 to 2 |

### 29.1.2 Register Base Address

Base addresses of TAPAn and PIC0 are listed in the following table.

Register addresses of TAPAn and PIC0 are given as offsets from the base addresses.

**Table 29.4 Register Base Addresses**

| Base Address Name | Base Address |
|---|---|
| <TAPA0_base> | FFE9 $0000_H$ |
| <PIC0_base> | FFDD $0000_H$ |

### 29.1.3 Clock Supply

The TAPAn and PIC0 clock supplies are listed in the following table.

**Table 29.5 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| TAPAn | PCLK | CKSCLK_IPERI1 | Module clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CKSCLK_IPERI1 | |
| PIC0 | PCLK | CKSCLK_IPERI1 | Module clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CKSCLK_IPERI1 | |

### 29.1.4 Interrupt Request

TAPA0 interrupt requests are listed in the following table.

**Table 29.6 Interrupt Requests**

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **TAPA0** | | | |
| TAPATIPEK0 | Peak interrupt 0 | 16, 116 | — |
| TAPATIVLY0 | Valley interrupt 0 | 17, 117 | — |

### 29.1.5 Reset Sources

Reset sources of TAPAn and PIC0 are listed in the following table. TAPAn and PIC0 are initialized by these reset sources.

**Table 29.7 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| TAPA0 | All reset sources (ISORES) |
| PIC0 | All reset sources (ISORES) |

### 29.1.6 External Input/Output Signal

External output signals of TAPAn and PIC0 are listed below.

**Table 29.8 External Input/Output Signals**

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **PIC** | | |
| TOUTU | Motor control output U phase (positive) | TAPA0UP |
| TOUTUB | Motor control output U phase (negative) | TAPA0UN |
| TOUTV | Motor control output V phase (positive) | TAPA0VP |
| TOUTVB | Motor control output V phase (negative) | TAPA0VN |
| TOUTW | Motor control output W phase (positive) | TAPA0WP |
| TOUTWB | Motor control output W phase (negative) | TAPA0WN |
| **TAPA** | | |
| TAPATHASIN | Motor control output Hi-Z control input | TAPA0ESO |

**CAUTION**

For the port pins that are used as TAPA0UP, TAPA0UN, TAPA0VP, TAPA0VN, TAPA0WP, and TAPA0WN, set the output driving ability to high (PDSCn_m = 1).

### 29.1.7 Internal Output Signal

The internal output signals of TAPAn and PIC0 are listed below.

**Table 29.9 Internal Output Signals**

| Unit Signal Name | Description | Connected to |
|---|---|---|
| **TAPA0** | | |
| $\overline{\text{TAPATHZOUT0}}$ | TAPA0UP/TAPA0UN output buffers Hi-Z control output*1 | Port |
| $\overline{\text{TAPATHZOUT1}}$ | TAPA0VP/TAPA0VN output buffers Hi-Z control output*1 | Port |
| $\overline{\text{TAPATHZOUT2}}$ | TAPA0WP/TAPA0WN output buffers Hi-Z control output*1 | Port |
| TAPATADOUT0 | A/D trigger signal 0 output*2 | ADCA0 hardware trigger expansion |
| TAPATADOUT1 | A/D trigger signal 1 output*2 | ADCA0 hardware trigger expansion |
| **PIC0** | | |
| TAPATHASIN | TAPA0 asynchronous Hi-Z control signal*1,*3 | TAPA0 |
| TAPATSIM0 | TAUD master channel interrupt signal<br>(TAUD0: INTTAUD0I0,INTTAUD0I2, INTTAUD0I8) | TAPA0 |
| TAPATUDCM0 | TAUD master up/down signal<br>(TAUD0: TAUD0UDC0, TAUD0UDC2, TAUD0UDC8) | TAPA0 |
| TAPATCDENS0 | TAUD slave 0 match detect*4<br>(ADCA0 hardware trigger expansion: ADOPA1ADCATTIN00) | TAPA0 |
| TAPATCDENS1 | TAUD slave 1match detect*4<br>(ADCA0 hardware trigger expansion: ADOPA2ADCATTIN00) | TAPA0 |

Note 1. See **Section 29.4.6, TAPA0 Hi-Z Control Input Selection** for details.

Note 2. These signals can be used to as a trigger source to start the A/D converter. See **Table 31.49, List of A/D Conversion Hardware Triggers**.

Note 3. This input signal is passed through a noise filter. See **Section 2.12, Noise Filter & Edge/Level Detector** and **Section 2.13, Description of Port Noise Filter & Edge/Level Detection**.

Note 4. These signals are selected by the H/W trigger selection bit of the ADCA0 A/D converter. See **Table 31.49, List of A/D Conversion Hardware Triggers**.

## 29.2 Overview

### 29.2.1 Functional Overview

The motor control function provides the following functions by combining the motor control unit (TAPA) and Timer Array Unit D (TAUDn) or A/D (ADCAn):

- Asynchronous Hi-Z control function
  Hi-Z control for TAUDn output can be performed by using pin input or error signals.
- Interrupt signal output function
  Request signals for two types of interrupts, peak interrupts and valley interrupts, can be output by the INTn signals output by TAUDn.
- A/D conversion start trigger selection function
  An A/D conversion start trigger can be output by the INTn signals output by TAUDn.

Additionally, the motor control function provides also the following functions by combining the motor control unit (TAPA) and the peripheral interconnection (PIC):

- Timer simultaneous start trigger function
  The respective channels of TAUD0 and TAUJ1, and the ENCAn timer can be started simultaneously.
- Trigger and pulse width measuring function
  Measurement of trigger periods can be performed by inputting ENCAn interrupt signals to TAUDn or TAUJ1.
- A/D trigger encoder capture function
  The value of the ENCAn counter can be captured at the A/D conversion start trigger timing.
- Three-phase PWM output with dead time / High-accuracy triangle PWM output with dead time
  Three-phase PWM output with dead time can be performed by TAUDn.
- Delay pulse output with dead time
  Delay pulses (with dead time) for the cycle timing can be output.

### 29.2.2 Basic Structure of Motor Control

The peripheral block configuration of the motor control function is shown below.

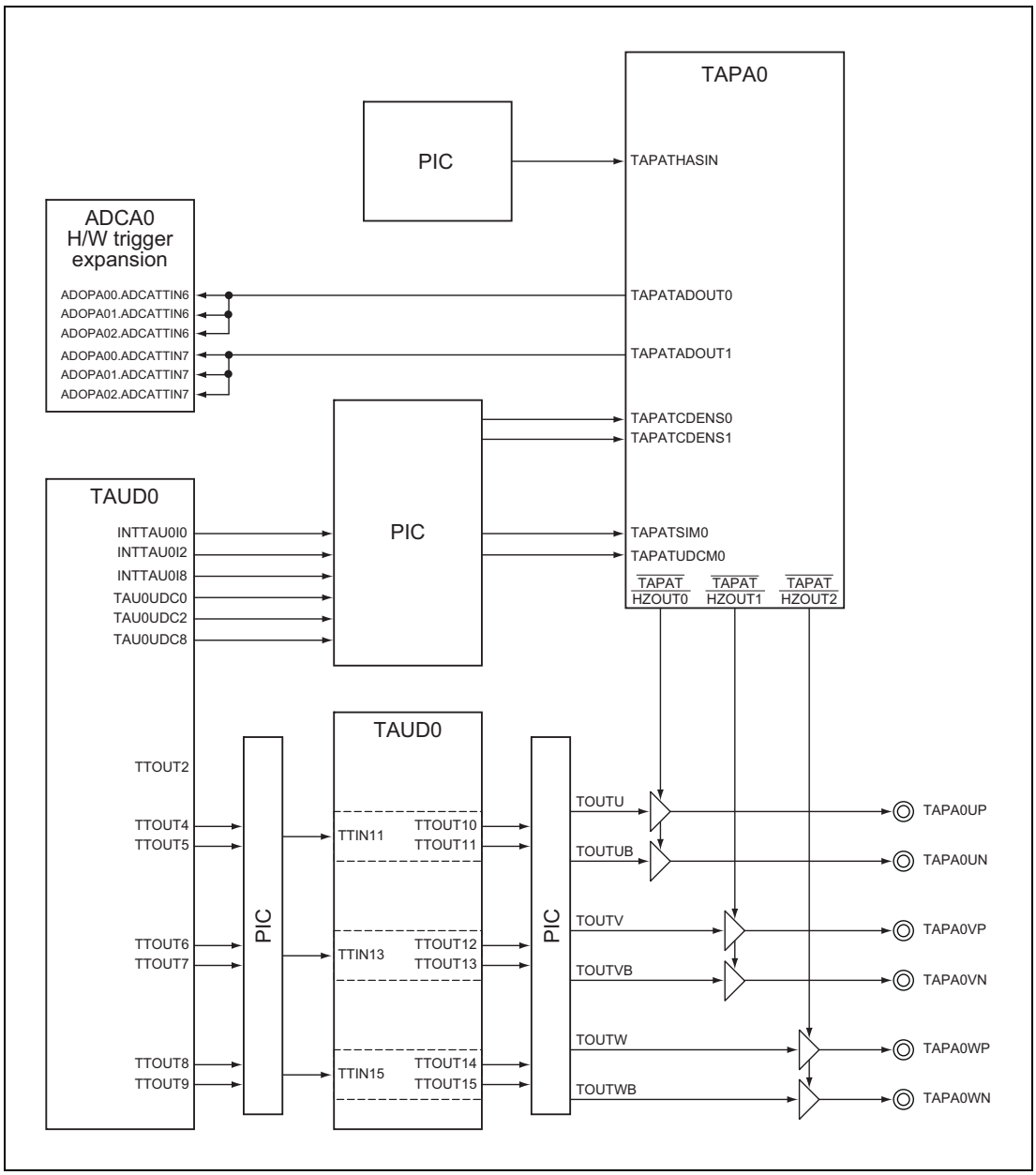



**Figure 29.1 Configuration of Motor Control**

TAUDn and PIC are used to generate the motor control output signals (three-phase PWM output signals with dead time).

The timer control unit (TAPA) performs Hi-Z control for the motor control output.

Additionally, PIC can provide functions specific to the motor by combining respective channels of TAUDn and TAUJ1, ENCAn, and TAPA.

### 29.2.3 Block Diagram

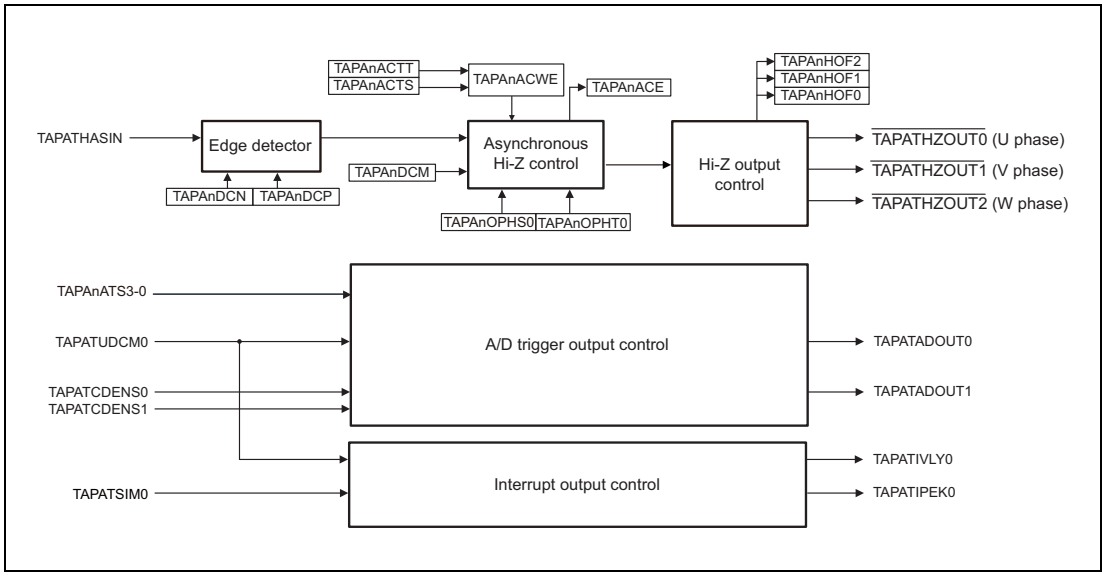



**Figure 29.2 TAPA Peripheral Block Diagram**

**NOTE**

For the PIC peripheral block diagram, see the respective section describing each function.

### 29.2.4 Definition of Terms

**Peak and valley interrupts - Peak and valley of timer counter**

In this document, the period from a TAUD counting-up status to generation of INT from the master channel is defined as a peak period, and this INT is defined as a peak interrupt.

In contrast, the period from a TAUD counting-down status to generation of INT from the master channel is defined as a valley period, and this INT is defined as a valley interrupt.

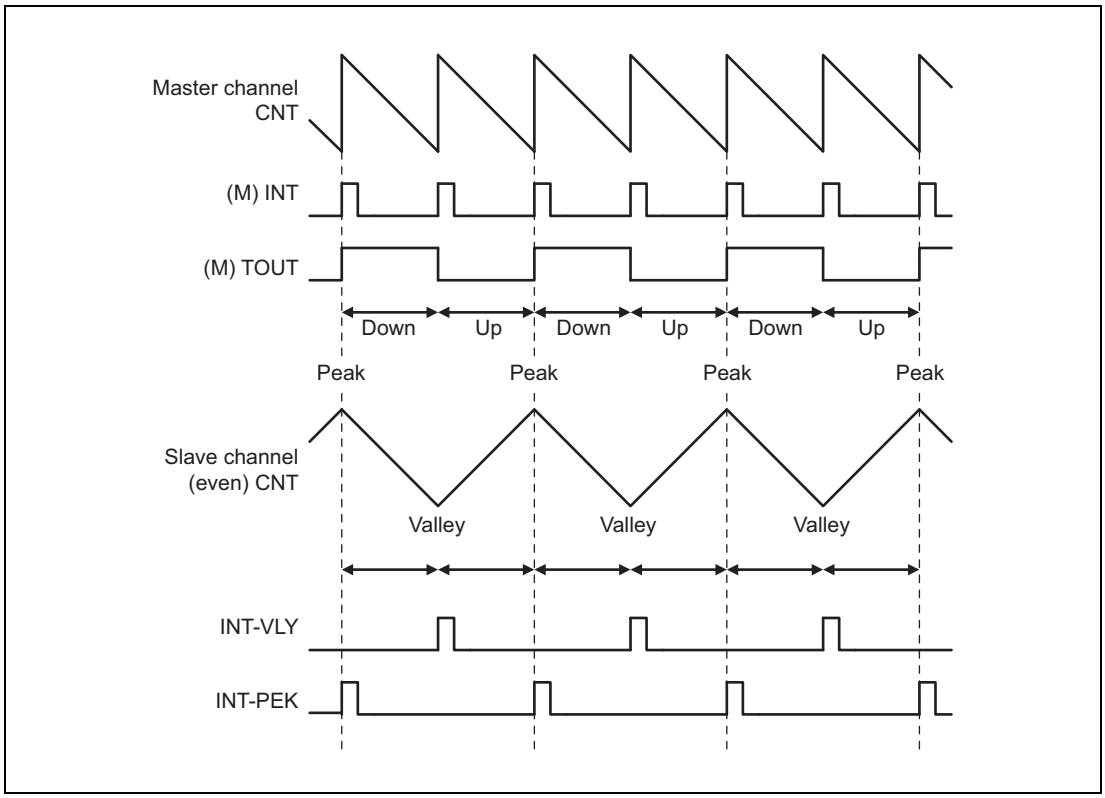



**Figure 29.3 Peak and Valley Interrupts**

# 29.3 Registers

## 29.3.1 List of Registers

The registers of TAPAn and PIC0 are listed in the following table.

For details about <TAPAn_base> and <PIC0_base>, see **Section 29.1.2, Register Base Address**.

**Table 29.10 Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| TAPAn | Control register 0 | TAPAnCTL0 | <TAPAn_base> + $20_H$ |
| | Control register 1 | TAPAnCTL1 | <TAPAn_base> + $24_H$ |
| | Flag register | TAPAnFLG | <TAPAn_base> + $00_H$ |
| | Asynchronous Hi-Z control write enable register | TAPAnACWE | <TAPAn_base> + $04_H$ |
| | Asynchronous Hi-Z control start trigger register | TAPAnACTS | <TAPAn_base> + $08_H$ |
| | Asynchronous Hi-Z control stop trigger register | TAPAnACTT | <TAPAn_base> + $0C_H$ |
| | Hi-Z start trigger register | TAPAnOPHS | <TAPAn_base> + $14_H$ |
| | Hi-Z stop trigger register | TAPAnOPHT | <TAPAn_base> + $18_H$ |
| | Emulation register | TAPAnEMU | <TAPAn_base> + $28_H$ |
| PIC0 | Simultaneous start trigger control register | PIC0SST | <PIC0_base> + $04_H$ |
| | Simultaneous start control register 0 | PIC0SSER0 | <PIC0_base> + $10_H$ |
| | Simultaneous start control register 2 | PIC0SSER2 | <PIC0_base> + $18_H$ |
| | Hi-Z output control register 0 | PIC0HIZCEN0 | <PIC0_base> + $80_H$ |
| | A/D conversion trigger output control register 400 | PIC0ADTEN400 | <PIC0_base> + $90_H$ |
| | A/D conversion trigger output control register 401 | PIC0ADTEN401 | <PIC0_base> + $94_H$ |
| | A/D conversion trigger output control register 402 | PIC0ADTEN402 | <PIC0_base> + $98_H$ |
| | Timer I/O control register 200 | PIC0REG200 | <PIC0_base> + $C0_H$ |
| | Timer I/O control register 201 | PIC0REG201 | <PIC0_base> + $C4_H$ |
| | Timer I/O control register 202 | PIC0REG202 | <PIC0_base> + $C8_H$ |
| | Timer I/O control register 203 | PIC0REG203 | <PIC0_base> + $CC_H$ |
| | Timer I/O control register 30 | PIC0REG30 | <PIC0_base> + $E8_H$ |
| | Timer I/O control register 31 | PIC0REG31 | <PIC0_base> + $EC_H$ |

**NOTE**

For details about PIC-related registers, see the respective section describing each function.

### 29.3.2 TAPAnCTL0 — TAPA Control Register 0

This register is used to set up the asynchronous Hi-Z control function.

The values of this register can be rewritten only when TAPAnFLG.TAPAnACE is 0 and TAUDnTEm for the corresponding TAUD's master channel is 0.

**Access:** This register can be read or written in 16-bit units.

**Address:** <TAPAn_base> + $20_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | TAPAn DCM | TAPAn DCN | TAPAn DCP | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R | R |

**Table 29.11 TAPAnCTL0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 5 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 4 | TAPAnDCM | Clear Condition Configuration<br>This control bit specifies the clear conditions for Hi-Z control output.<br>0: Enables manipulation of TAPAnOPHT0 regardless of the TAPATHASIN signal input level.<br>1: Enables manipulation of TAPAnOPHT0 only if the TAPATHASIN signal input is inactive. |
| 3, 2 | TAPAnDCN, TAPAnDCP | Hi-Z Input Edge Selection<br>These are control bits that specify the valid edge of TAPATHASIN. |

| TAPAn DCN | TAPAn DCP | Description |
|---|---|---|
| 0 | 0 | Does not detect valid edges. |
| 0 | 1 | Detects a rising edge as the valid edge (active level = high). |
| 1 | 0 | Detects a falling edge as the valid edge (active level = low). |
| 1 | 1 | Setting prohibited |

| | | |
|---|---|---|
| 1, 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

### 29.3.3 TAPAnCTL1 — TAPA Control Register 1

This register is used to specify the A/D conversion trigger.

**Access:** This register can be read or written in 8-bit units.

**Address:** <TAPAn_base> + $24_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TAPAnATS3 | TAPAnATS2 | TAPAnATS1 | TAPAnATS0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 29.12 TAPAnCTL1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3, 2 | TAPAnATS3, TAPAnATS2 | A/D Conversion Trigger 1 Selection<br>These are control bits that specify A/D conversion trigger output 1 (TAPATADOUT1). |
| 1, 0 | TAPAnATS1, TAPAnATS0 | A/D Conversion Trigger 0 Selection<br>These are control bits that specify the A/D conversion trigger output 0 (TAPATADOUT0). |

| TAPAn ATS3 | TAPAn ATS2 | Description |
|---|---|---|
| 0 | 0 | INT signal while the triangle wave is falling (counting down) |
| 0 | 1 | INT signal while the triangle wave is rising (counting up) |
| 1 | 0 | INT signal while the triangle wave is rising (counting up) or falling (counting down) |
| 1 | 1 | INT signal while the triangle wave is rising (counting up) or falling (counting down) and valley interrupt TAPATIVLY0 signal |

| TAPAn ATS1 | TAPAn ATS0 | Description |
|---|---|---|
| 0 | 0 | INT signal while the triangle wave is falling (counting down) |
| 0 | 1 | INT signal while the triangle wave is rising (counting up) |
| 1 | 0 | INT signal while the triangle wave is rising (counting up) or falling (counting down) |
| 1 | 1 | INT signal while the triangle wave is rising (counting up) or falling (counting down) and valley interrupt TAPATIVLY0 signal |

### 29.3.4 TAPAnFLG — TAPA Flag Register

This flag register is for asynchronous Hi-Z control.

**Access:** This register is a read-only register that can be read in 16-bit units.

**Address:** <TAPAn_base> + $00_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TAPAn HOF2 | TAPAn HOF1 | TAPAn HOF0 | — | — | — | — | — | — | — | TAPAn ACE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 29.13 TAPAnFLG Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 11 | Reserved | When read, the value after reset is returned. |
| 10 | TAPAnHOF2 | W phase Hi-Z Control Monitor<br>This bit is used to monitor the Hi-Z control status.<br>0: The present output of TAPAnTHZOUT2 is high level<br>1: The present output TAPAnTHZOUT2 is low level. |
| 9 | TAPAnHOF1 | V phase Hi-Z Control Monitor<br>This bit is used to monitor the Hi-Z control status.<br>0: The present output of TAPAnTHZOUT1 is high level<br>1: The present output TAPAnTHZOUT1 is low level. |
| 8 | TAPAnHOF0 | U phase Hi-Z Control Monitor<br>This bit is used to monitor the Hi-Z control status.<br>0: The present output of TAPAnTHZOUT0 is high level<br>1: The present output TAPAnTHZOUT0 is low level. |
| 7 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | TAPAnACE | Asynchronous Hi-Z Control Enable<br>0: Indicates that the asynchronous Hi-Z control is stopped.<br>1: Indicates that the asynchronous Hi-Z control is enabled.<br>The conditions for setting or clearing this bit are as follows:<br>Clear condition: Writing 1 to TAPAnACTT while TAPAnACWE = 1<br>Set condition: Writing 1 to TAPAnACTS while TAPAnACWE = 1 |

### 29.3.5 TAPAnACWE — TAPA Asynchronous Hi-Z Control Write Enable Register

This register is used to enable writing for asynchronous Hi-Z control.

**Access:** This register can be read or written in 8-bit units.

**Address:** <TAPAn_base> + $04_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | TAPAnACWE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 29.14 TAPAnACWE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | TAPAnACWE | Asynchronous Control Write Enable<br>This is a write enable bit for asynchronous Hi-Z control.<br>After 1 is written, this bit is automatically cleared to 0 by writing 1 to TAPAnACTS and TAPAnACTT.<br>0: Disables writing to TAPAnACTS and TAPAnACTT.<br>1: Enables writing to TAPAnACTS and TAPAnACTT. |

### 29.3.6 TAPAnACTS — TAPA Asynchronous Hi-Z Control Start Trigger Register

This register is used to enable the start trigger for asynchronous Hi-Z control.

**Access:** This register is a write-only register that can be written in 8-bit units. This register is always read as $00_H$.

**Address:** <TAPAn_base> + $08_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | TAPAnACTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 29.15 TAPAnACTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | TAPAnACTS | Asynchronous Hi-Z Control Start Trigger<br>This bit enables the start trigger for asynchronous Hi-Z control.<br>The setting of this bit is valid only when TAPAnACWE = 1.<br>0: Writing 0 is ignored (no function).<br>1: Enables asynchronous Hi-Z control when TAPAnACE is 1. |

## 29.3.7 TAPAnACTT — TAPA Asynchronous Hi-Z Control Stop Trigger Register

This bit enables the stop trigger for asynchronous Hi-Z control.

**Access:** This register is a write-only register that can be written in 8-bit units. This register is always read as $00_H$.

**Address:** <TAPAn_base> + $0C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | TAPAnACTT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 29.16 TAPAnACTT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | TAPAnACTT | Asynchronous Hi-Z Control Stop Trigger<br>This bit enables the stop trigger for asynchronous Hi-Z control.<br>The setting of this bit is valid only when TAPAnACWE = 1.<br>0: Writing 0 is ignored (no function).<br>1: Disables asynchronous Hi-Z control when TAPAnACE is 0. |

## 29.3.8 TAPAnOPHS — TAPA Hi-Z Start Trigger Register

This software trigger register is used to start Hi-Z control for motor control output pins.

**Access:** This register is a write-only register that can be written in 8-bit units. This register is always read as $00_H$.

**Address:** <TAPAn_base> + $14_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | TAPAnOPHS0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 29.17 TAPAnOPHS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | TAPAnOPHS0 | Hi-Z Control Start Trigger<br>This bit starts Hi-Z control for motor control output pins.<br>0: Writing 0 is ignored (no function).<br>1: Starts Hi-Z control. |

### 29.3.9 TAPAnOPHT — TAPA Hi-Z Stop Trigger Register

This software trigger register is used to stop Hi-Z control for motor control output pins.

**Access:** This register is a write-only register that can be written in 8-bit units. This register is always read as $00_H$.

**Address:** <TAPAn_base> + $18_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | TAPAnOPHT0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 29.18 TAPAnOPHT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | TAPAnOPHT0 | Hi-Z Control Stop Trigger<br>This bit stops Hi-Z control for motor control output pins.<br>0: Writing 0 is ignored (no function).<br>1: Stops Hi-Z control. Whether the setting of this bit is valid or invalid depends on the setting of TAPAnCTL0.TAPAnDCM. |

### 29.3.10 TAPAnEMU — TAPA Emulation Register

This register controls SVSTOP for emulation.

**Access:** This register can be read or written in 8-bit units. (when EPC.SVSTOP = 0, rewritten only)

**Address:** <TAPAn_base> + $28_H$

**Value after reset:** Reading this register returns always $00_H$.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TAPAnSVSDIS | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R |

**Table 29.19 TAPAnEMU Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | TAPAnSVSDIS | This bit is used to control disabling of SVSTOP.<br>0: SVSTOP is valid. (Sets Hi-Z control output to low level when SVSTOP = 1 is input).<br>1: SVSTOP is invalid. (Hi-Z control output level does not change according to the level of SVSTOP input). |
| 6 to 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

## 29.4 Asynchronous Hi-Z Control Function

If the operation of the timer motor control function controlled by the MCU becomes abnormal, the rotation of the external motor also becomes abnormal. This function can forcibly set the motor control output to the Hi-Z state upon detection of abnormal motor operation, independent of MCU control.

### 29.4.1 Overview

This function forcibly stops TAPAn output through asynchronous Hi-Z control.

- When the TAPATHASIN signal becomes active, the levels of the motor control output pins are set to Hi-Z, and motor control output is forcibly stopped.
- Motor control output in a Hi-Z state can be resumed by writing the Hi-Z stop trigger register (TAPAnOPHT0).
- The Hi-Z state of motor control output can also be specified by writing the Hi-Z control start trigger register (TAPAnOPHS).
- Setting PIC can enable or disable Hi-Z control input when an error occurs.

### 29.4.2 System Configuration Example

A system configuration example is shown below, where an external error detection signal (the TAPA0ESO signal) is used for Hi-Z control of the motor control outputs (TAPA0UP / TAPA0UN / TAPA0VP / TAPA0VN / TAPA0WP / TAPA0WN).

When valid edges of the external error detection signal are detected, the level of the motor control outputs is set to Hi-Z.

Because the microcontroller might freeze when an error occurs, external error detection signals are continuously processed so that the motor control timer outputs can be set to Hi-Z even if no clock is supplied.

Note that an error is detected only when the valid edge of the error detection signal is detected. Therefore, no error is detected if the output level is fixed and the signal level does not change.

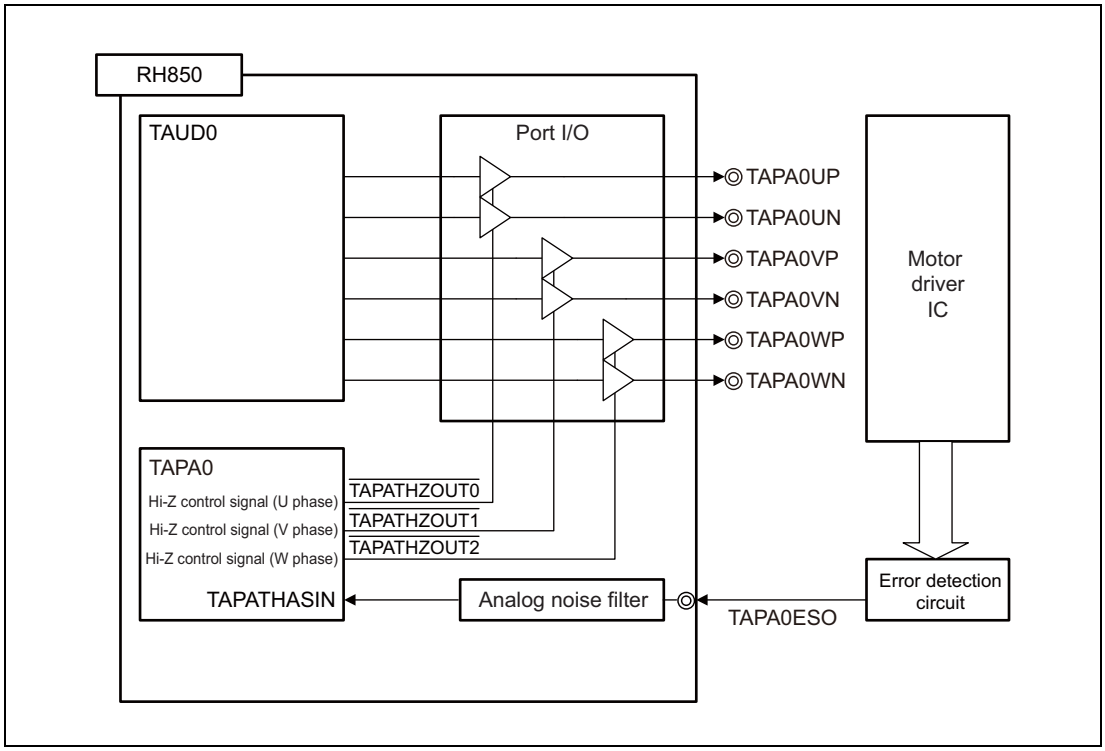



**Figure 29.4 System Configuration Example of Asynchronous Hi-Z Control for Pin Input**

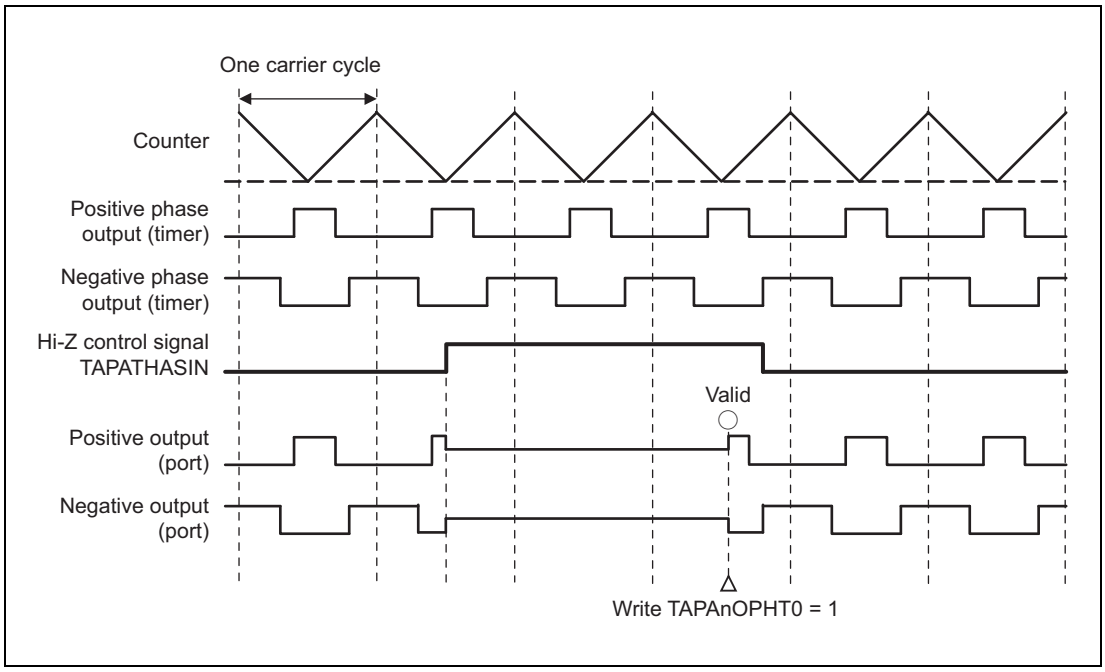
### 29.4.3 Basic Operation

Hi-Z control for motor control output pins can be started as follows:

- Detecting the valid edge of asynchronous Hi-Z control signal (TAPATHASIN)
- Setting the start trigger bit TAPAnOPHS.TAPAnOPHS0 of the Hi-Z control signal

The levels of the motor control output pins are set to Hi-Z until the stop trigger bit of the Hi-Z control signal (TAPAnOPHT.TAPAnOPHT0) is set. Note that whether the setting of TAPAnOPTH0 is valid or invalid depends on the setting of TAPAnCTL0.TAPAnDCM.

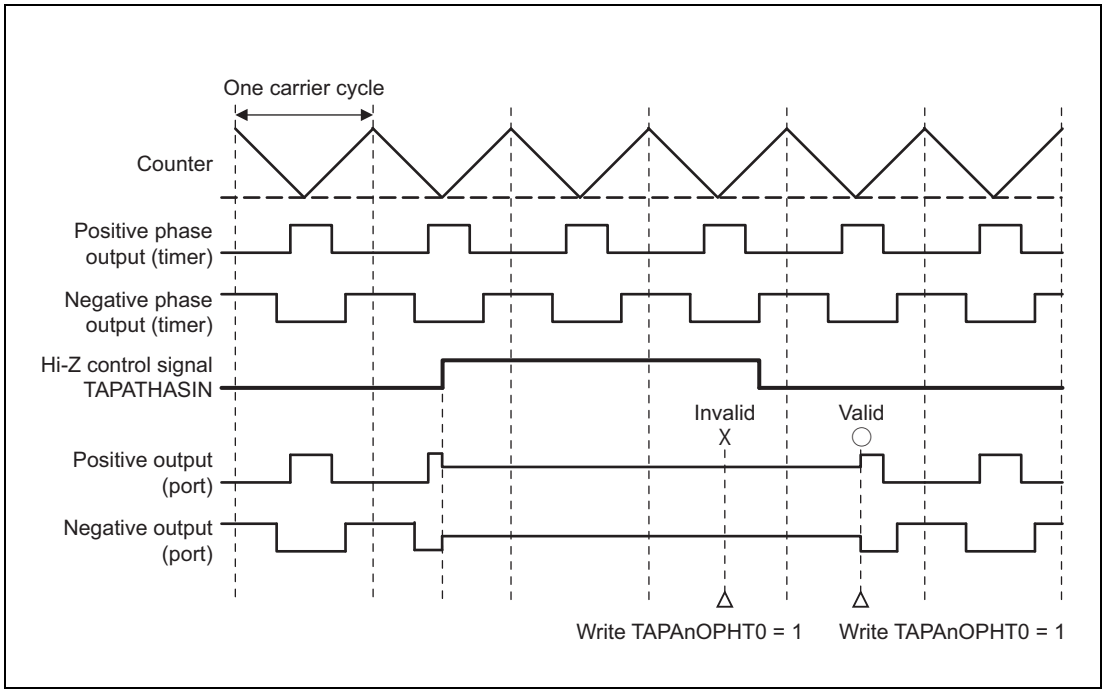
#### (1) Operation when TAPAnCTL0.TAPAnDCM = 0, TAPAnDCP = 1, and TAPAnDCN = 0




The motor control outputs are forcibly stopped (Hi-Z output) when the valid edge of TAPATHASIN is detected.

The motor control outputs restart when 1 is written to TAPAnOPHT.TAPAnOPHT0, regardless of the level of TAPATHASIN.

### (2) Operation when TAPAnCTL0.TAPAnDCM = 1, TAPAnDCP = 1, and TAPAnDCN = 0




The motor control outputs are forcibly stopped (Hi-Z output) when the valid edge of TAPATHASIN is detected.

Writing 1 to the stop trigger bit (TAPAnOPHT.TAPAnOPHT0) of the Hi-Z control signal is ignored while TAPATHASIN is active (high level because TAPAnCTL0.TAPAnDCP is 1).

The motor control outputs restart when 1 is written to TAPAnOPHT.TAPAnOPHT0 after TAPATHASIN becomes inactive (low level because TAPAnCTL0.TAPAnDCP is 1).

### 29.4.4 Asynchronous Hi-Z Control Using Software Trigger

Hi-Z control for motor control output is possible by using the Hi-Z control start trigger bit TAPAnOPHS.TAPAnOPHS0 and Hi-Z control stop trigger bit TAPAnOPHT.TAPAnOPHT0.

#### (1) Function of Hi-Z control start trigger bit TAPAnOPHS.TAPAnOPHS0

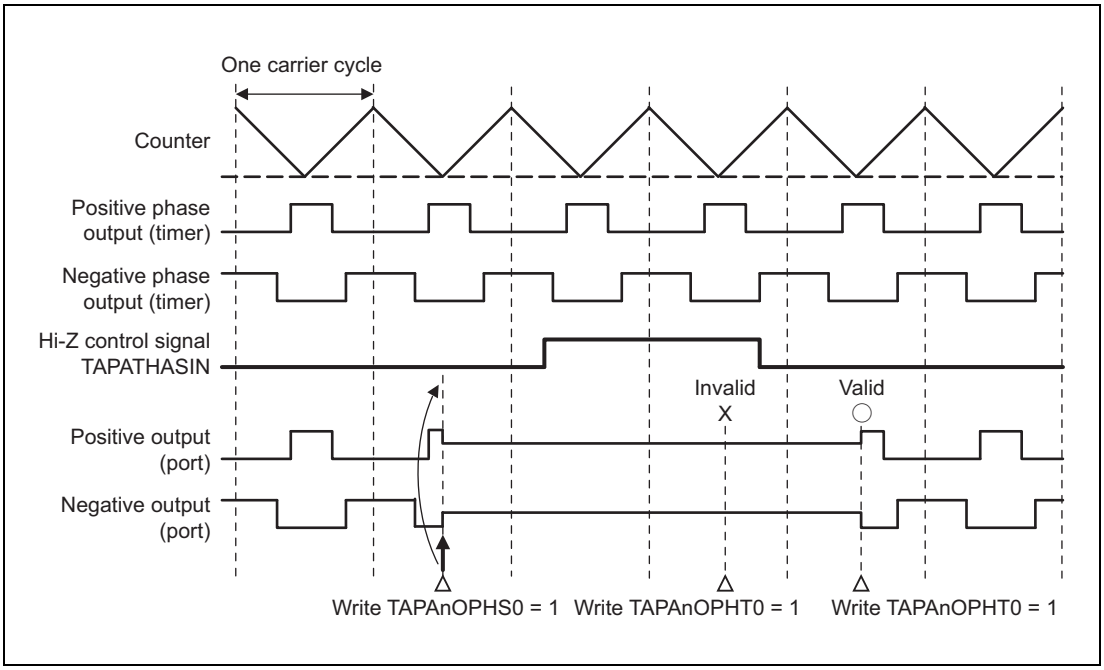
| TAPAnDCM | Function |
|---|---|
| 0/1 | Writing 1 to TAPAnOPHS0 starts Hi-Z control and forcibly stops the motor control output (Hi-Z output). |

#### (2) Function of Hi-Z control stop trigger bit TAPAnOPHT.TAPAnOPHT0

Whether the Hi-Z control stop trigger is valid or invalid depends on the conditions below:

| TAPAnDCM | Function |
|---|---|
| 0 | Writing 1 to TAPAnOPHT0 stops Hi-Z control and restarts motor control output. |
| 1 | If TAPATHASIN is inactive, writing 1 to TAPAnOPHT0 stops Hi-Z control and restarts motor control output.<br>If TAPATHASIN is active, writing 1 to TAPAnOPHT0 is ignored. |

#### (3) Operation when TAPAnCTL0.TAPAnDCM = 1, TAPAnDCP = 1, and TAPAnDCN = 0




The motor control output (Hi-Z output) is forcibly stopped when 1 is written to TAPAnOPHS0.

After that, the level of the motor control output remains Hi-Z even if the rising edge of TAPATHASIN is detected.

Writing to TAPAnOPHT0 is ignored while TAPATHASIN is active (high level because TAPAnDCN is 0 and TAPAnDCP is 1).

After detection of the falling edge of TAPATHASIN, the motor control output restarts when 1 is written to TAPAnOPHT0 while TAPATHASIN is inactive (low level because TAPAnDCN is 0 and TAPAnDCP is 1).

### 29.4.5 Operating Procedure

The operating procedure for the asynchronous input Hi-Z control function is shown below:

| | Operation | Status of TAPA |
|---|---|---|
| Initial setup | Set up the TAPAnCTL0 register.<br>Specify TAPAnDCP and TAPAnDCN to select the input edge.<br>Specify TAPAnDCM to select the clear mode. | Asynchronous Hi-Z control stopped<br>(TAPAnFLG.TAPAnACE = 0) |
| Start operation | Set up the TAPAnACWE register.<br>Set TAPAnACWE to 1.<br><br>Set up the TAPAnACTS register.<br>Set TAPAnACTS to 1. | <br>Writing to TAPAnACTS is enabled.<br><br>Asynchronous Hi-Z control enabled<br>(TAPAnFLG.TAPAnACE = 1) |
| During operation | Hi-Z control for the timer function outputs can be started by controlling the following:<br>• TAPAnOPHS register<br>• Asynchronous Hi-Z control signal (TAPATHASIN)<br><br>Hi-Z control for the timer function outputs can be stopped by controlling the following:<br>• TAPAnOPHT register<br>(If TAPAnDCM is 1, control by the TAPAnOPHT register is enabled only while TAPATHASIN is inactive.)<br><br>The TAPA operating status can always be read using the TAPAnFLG register. | Hi-Z control for the motor control output pins is started by detecting the valid edge of the asynchronous Hi-Z control signal (TAPATHASIN) or by setting the Hi-Z control start trigger bit TAPAnOPHS0 to 1.<br>Hi-Z control for the motor control output pins is stopped by setting the Hi-Z control stop trigger bit TAPAnOPHT0 to 1 according to the operation mode specified by the TAPAnDCM bit. |
| Stop operation | Set up the TAPAnACWE register.<br>Set TAPAnACWE to 1.<br><br>Set up the TAPAnACTT register.<br>Set TAPAnACTT to 1. | <br>Writing to TAPAnACTT is enabled.<br><br>Asynchronous Hi-Z control stopped<br>(TAPAnFLG.TAPAnACE = 0) |

Restart

### 29.4.6 TAPA0 Hi-Z Control Input Selection

In order to stop the motor control output in case of errors, error events are selected in PIC and the level of the motor control output is set to Hi-Z in TAPA0, as shown in the diagram below.

The TAPA function can be stopped by setting TAPA0ACTT = $01_H$ after setting PIC0HIZCEN0 = $00_H$ or TAPA0ACWE = $01_H$.

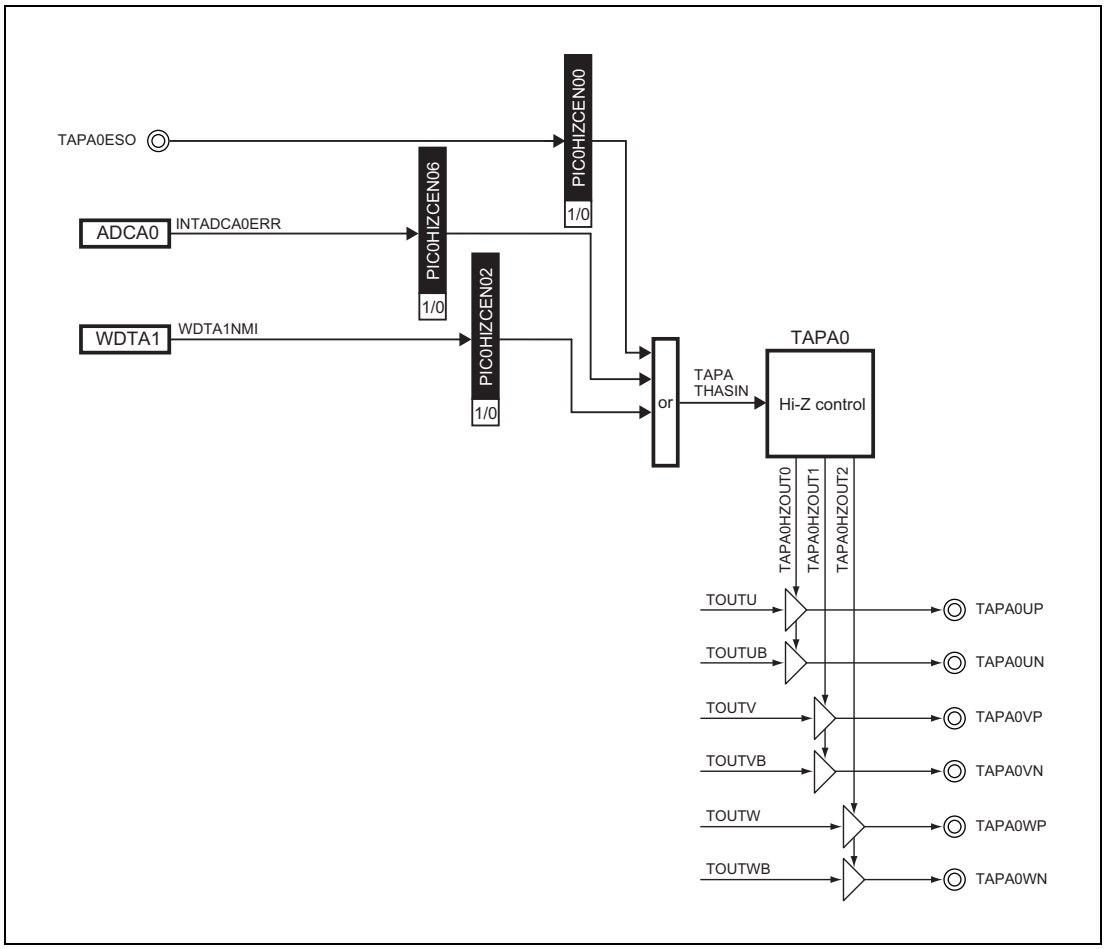



**Figure 29.5 Hi-Z Control Block Diagram**

Switching into a Hi-Z state can be performed by the following:

- TAPA0ESO pin input
- A/D converter ADCA0 error signal ADCA0ERR
- Window Watchdog Timer WDTA1 non maskable interrupt WDTA1NMI

For details about these signals, see the respective descriptions.

## 29.4.7 Registers

### 29.4.7.1 PIC0HIZCENn — Hi-Z Output Control Register n (n = 0)

The PIC0HIZCENn register selects the Hi-Z output control signal of TAPAn.

**Access:** This register can be read or written in 8-bit units.

**Address:** <PIC0_base> + $80_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | PIC0HIZCENn6 | — | — | — | PIC0HIZCENn2 | — | PIC0HIZCENn0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R | R | R | R/W | R | R/W |

**Table 29.20 PIC0HIZCENn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 6 | PIC0HIZCENn6 | Selects whether to enable or disable Hi-Z output control by the INTADCA0ERR interrupt signal.<br>0: Disable<br>1: Enable |
| 5 to 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 | PIC0HIZCENn2 | Selects whether to enable or disable Hi-Z output control by the WDTA1NMI interrupt signal.<br>0: Disable<br>1: Enable |
| 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | PIC0HIZCENn0 | Selects whether to enable or disable Hi-Z output control by the TAPA0ESO pin input.<br>0: Disable<br>1: Enable |

# 29.5 INT Signal Output Selection Function

## 29.5.1 Configuration of the INT Signal Output Selection Function

This function generates the peak interrupt TAPATIPEK0 and valley interrupt TAPATIVLY0 by using the TAPATSIM0 signal, which is connected to the INT signal on the TAUD's triangular carrier cycle generation channel (master) and TAPATUDCM0 signal, which is connected to the counter up/down signal.

For the connection destination of TAPATSIM0, see **Section 29.1.7, Internal Output Signal**.

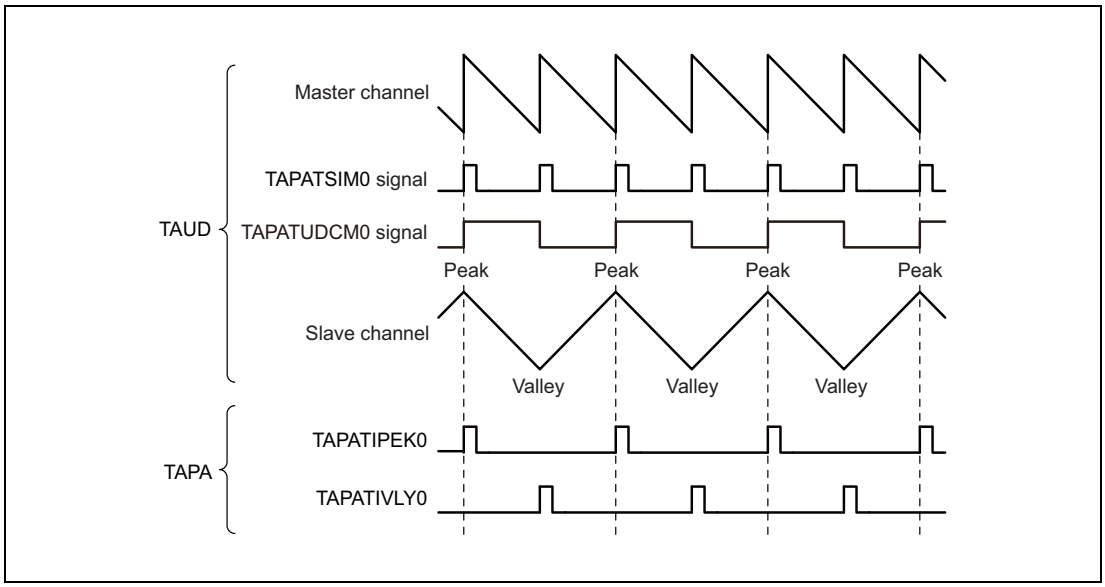



**Figure 29.6 Basic Timing of Signals for the INT Signal Output Selection Function**

Triangular carrier cycles are generated on the master channel.

The INT signal generated on the master channel in each half triangular carrier cycle is input to TAPAn as the TAPATSIM0 signal. TAPAn generates the TAPATIPEK0 signal (peak interrupt) during high level of the TAPATUDCM0 signal and the TAPATIVLY0 signal (valley interrupt) during low level of the TAPATUDCM0 signal by using the TAPATSIM0 and TAPATUDCM0 input signals.

**CAUTION**

**The peak interrupt TAPATIPEK0 and valley interrupt TAPATIVLY0 are generated regardless of the function of the master channel of TAUD.**

**When not using these peak and valley interrupts, mask them by using the ICTAPAnIPEK0 and ICTAPAnIVLY0 registers, respectively.**

### 29.5.2 Block Diagram

TAUDn and TAPAn are connected in the registers shown below by the INT signal output selection function.

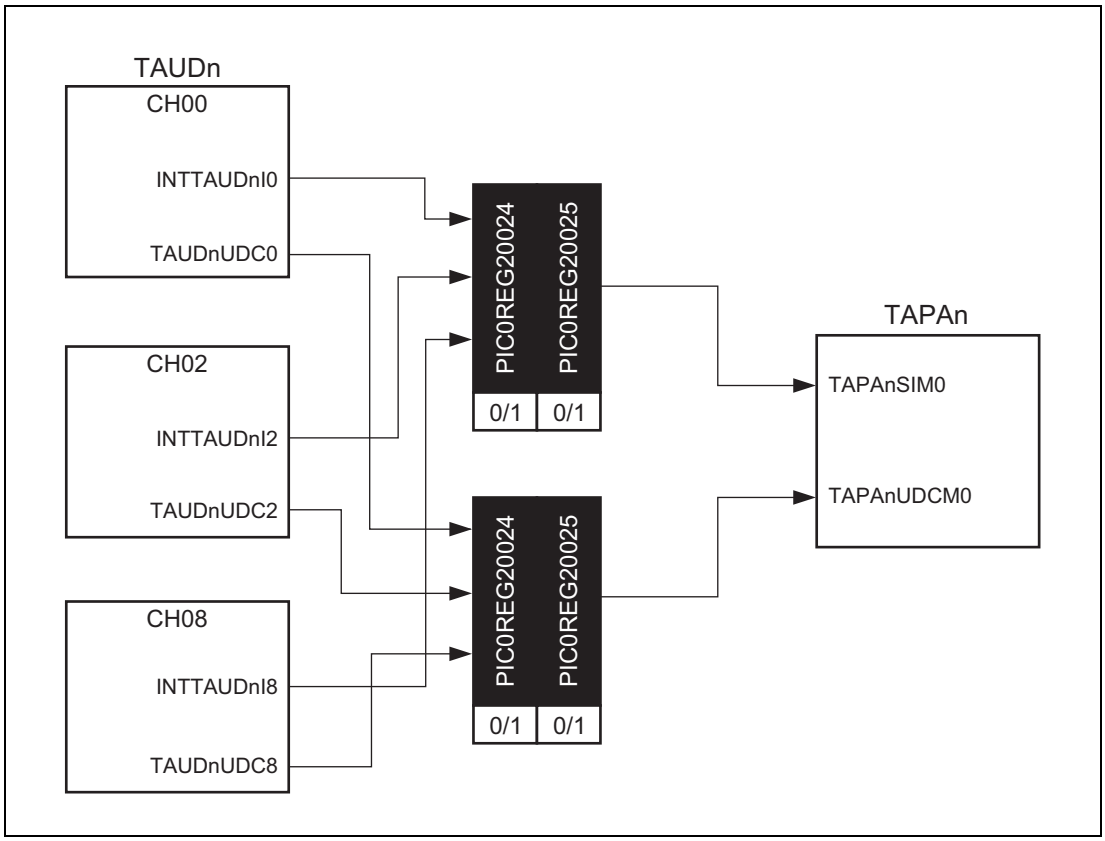



**Figure 29.7 Connection of the INT Signals**

## 29.5.3 Registers

### 29.5.3.1 PIC0REG2n0 — Timer I/O Control Register 2n0 (n = 0)

This register selects the TAPA0 input.

**Access:** This register can be read or written in 32-bit units.

**Address:** PIC0REG200: FFDD 00C0$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | PIC0REG2n025 | PIC0REG2n024 | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 29.21 PIC0REG2n0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | *1 |
| 25, 24 | PIC0REG2n025<br>PIC0REG2n024 | Select the TAUDn channel used by TAPATSIM0 and TAPATUDCM0.<br>00: Not selected<br>01: TAUD0 channel 0 selected<br>10: TAUD0 channel 2 selected<br>11: TAUD0 channel 8 selected |
| 23 to 0 | Reserved | *1 |

Note 1. Some of the bits defined as 0 in the PIC0REG2n0 register are defined for the other timer connection functions. For such bits, use the bit definition of those timer connection functions.

## 29.6 A/D Converter Conversion Trigger Selection Function

This function outputs the A/D converter conversion trigger signals TAPATADOUT0 and TAPATADOUT1 from the TAPATCDENS0 and TAPATCDENS1 signals, which are connected to a compare match interrupt based on the triangular carrier cycle of TAUD, or a valley interrupt signal (TAPATIVLY0).

### 29.6.1 Configuration of A/D Converter Conversion Trigger Selection Function

**Table 29.22 Signals Used for TAPATADOUT Generation**

| Output Signal | Slave Match Detection Signal | Valley Interrupt Signal |
|---|---|---|
| TAPATADOUT0 | TAPATCDENS0 | TAPATIVLY0 |
| TAPATADOUT1 | TAPATCDENS1 | TAPATIVLY0 |

**Table 29.23 Operation of TAPATADOUT1 According to the Setting of TAPAnCTL1.TAPAnATS[3:2]**

| TAPAnATS3 | TAPAnATS2 | Description |
|---|---|---|
| 0 | 0 | Outputs the INT signal from TAPATADOUT1 while the triangle wave is falling (counting down). |
| 0 | 1 | Outputs the INT signal from TAPATADOUT1 while the triangle wave is rising (counting up). |
| 1 | 0 | Outputs the INT signal from TAPATADOUT1 while the triangle wave is rising (counting up) or falling (counting down). |
| 1 | 1 | Outputs the INT signal while the triangle wave is rising (counting up) or falling (counting down) and valley interrupt TAPATIVLY0 from TAPATADOUT1. |

**Table 29.24 Operation of TAPATADOUT0 According to the Setting of TAPAnCTL1.TAPAnATS[1:0]**

| TAPAnATS1 | TAPAnATS0 | Description |
|---|---|---|
| 0 | 0 | Outputs the INT signal from TAPATADOUT0 while the triangle wave is falling (counting down). |
| 0 | 1 | Outputs the INT signal from TAPATADOUT0 while the triangle wave is rising (counting up). |
| 1 | 0 | Outputs the INT signal from TAPATADOUT0 while the triangle wave is rising (counting up) or falling (counting down). |
| 1 | 1 | Outputs the INT signal while the triangle wave is rising (counting up) or falling (counting down) and valley interrupt TAPATIVLY0 from TAPATADOUT0. |

## 29.6.2 Block Diagram

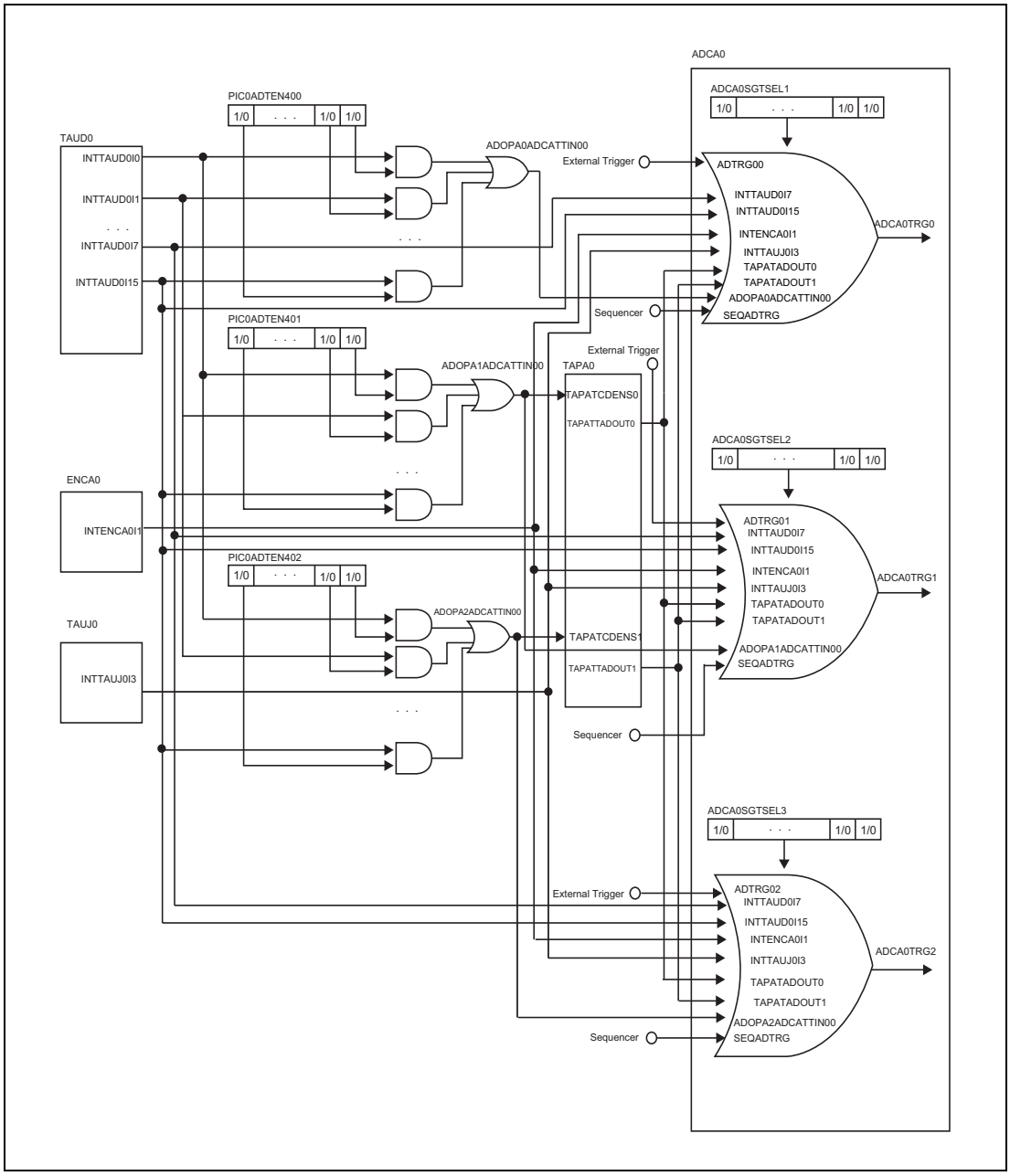



**Figure 29.8 Block Diagram of A/D Conversion Trigger Selection Function**

**NOTE**

See **Section 31.3.4.1, ADCAnSGTSELx — Scan Group x Start Trigger Control Register x** for details on the settings of the ADCA0SGTSEL register.

### 29.6.3 Waveforms of A/D Converter Conversion Trigger Output Control Operation in Triangle PWM Mode

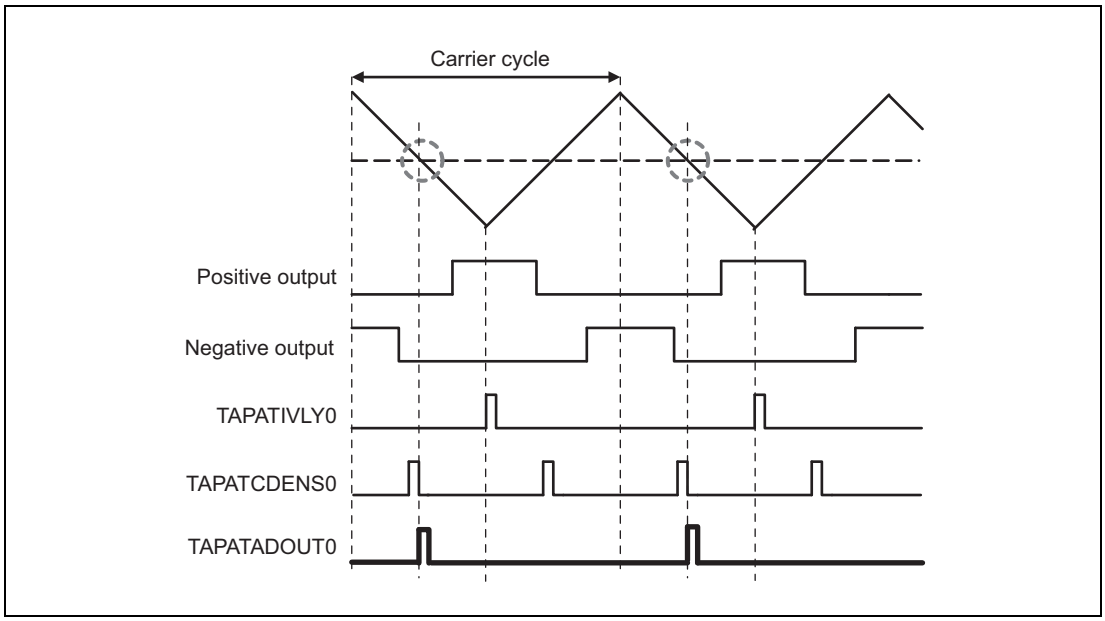



**Figure 29.9 TAPAnATS[1:0] bits = $00_B$: Output INT Signal while the Triangle Wave is Falling (Counting Down)**

While the triangle wave is falling (counting down), the signals TAPATCDENS0 and TAPATCDENS1 are output as the A/D converter conversion trigger signals TAPATADOUT0 and TAPATADOUT1.

In this case, no A/D converter conversion trigger signal is output while the triangle wave is rising (counting up).

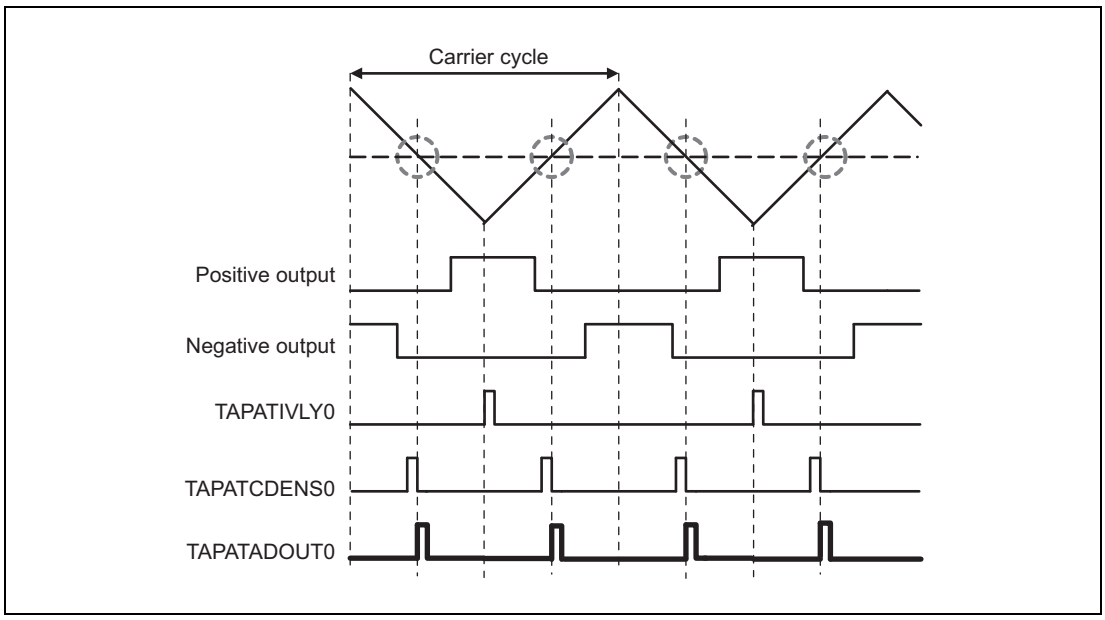



**Figure 29.10 TAPAnATS[1:0] bits = $10_B$: Output INT Signal while the Triangle Wave is Rising (Counting Up) or Falling (Counting Down)**

The signals TAPATCDENS0 and TAPATCDENS1 are output as the A/D converter conversion trigger signals TAPATADOUT0 and TAPATADOUT1.

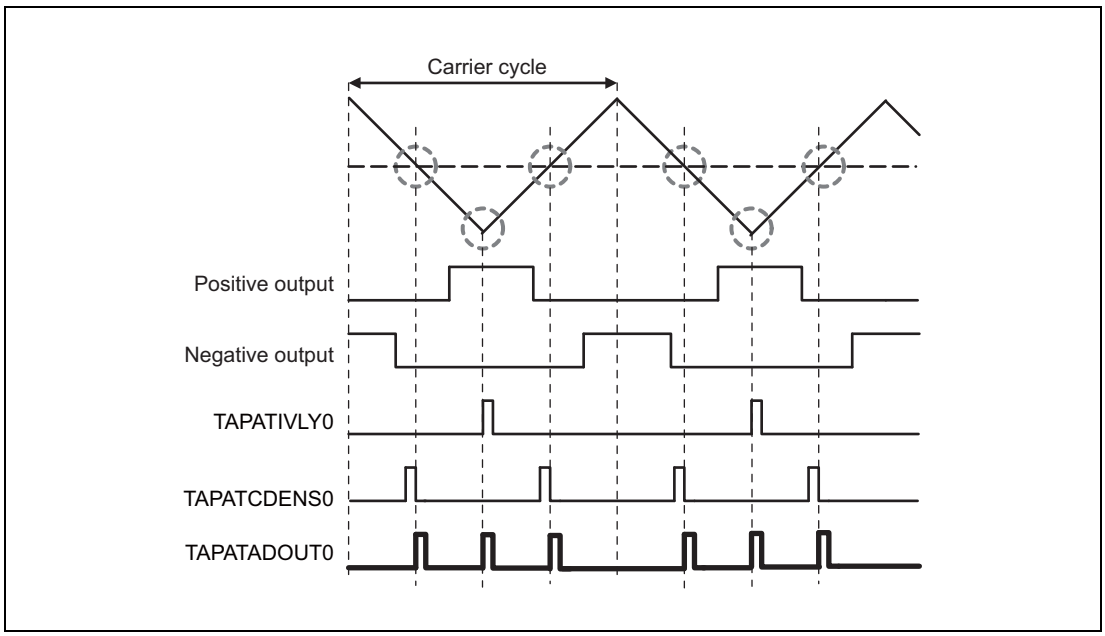



**Figure 29.11 TAPAnATS[1:0] bits = $11_B$: Output of INT Signal and Valley Interrupt while the Triangle Wave is Rising (Counting Up) or Falling (Counting Down)**

The signals TAPATCDENS0 and TAPATCDENS1 and valley interrupt TAPATIVLY0 are output as the A/D converter conversion trigger signals TAPATADOUT0 and TAPATADOUT1.

### 29.6.4 Operating Procedure for A/D Converter Conversion Trigger Selection Function

The operating procedure for the A/D converter conversion trigger selection function is shown below.

| | Operation | Status of TAUD and TAPA |
|---|---|---|
| Initial setup | Initialize TAUD.<br>Specify the timer operation mode.<br><br>Set up the TAPAnCTL1 register.<br>Specify TAPAnATS[1:0] (TAPATADOUT0 setting).<br>Specify TAPAnATS[3:2] (TAPATADOUT1 setting).<br><br>Set up the PIC0ADTEN4nj and PIC0REG2n0 registers according to the signal to be used.<br>Specify PIC0ADTEN4nj (TAPATCDENS0 or TAPATCDENS1 setting).<br>Specify PIC0REG2n0 (TAPATIVLY0 setting). | TAUD and TAPA stop the operation. |
| Start operation | Start the TAUD operation. | TAUD starts the count operation. |
| During operation | TAUD operates according to the setting of each function. | The A/D conversion trigger selection function outputs either TAPATADOUT0 according to the setting of TAPAnATS[1:0] or TAPATADOUT1 according to the setting of TAPAnATS[3:2], based on the interrupt TAPATCDENS1 or TAPATCDENS0, which is input from TAUD, and the valley interrupt TAPATIVLY0, which is generated by TAPA. |
| Stop operation | Stop the TAUD operation. | TAUD stops the count operation. |

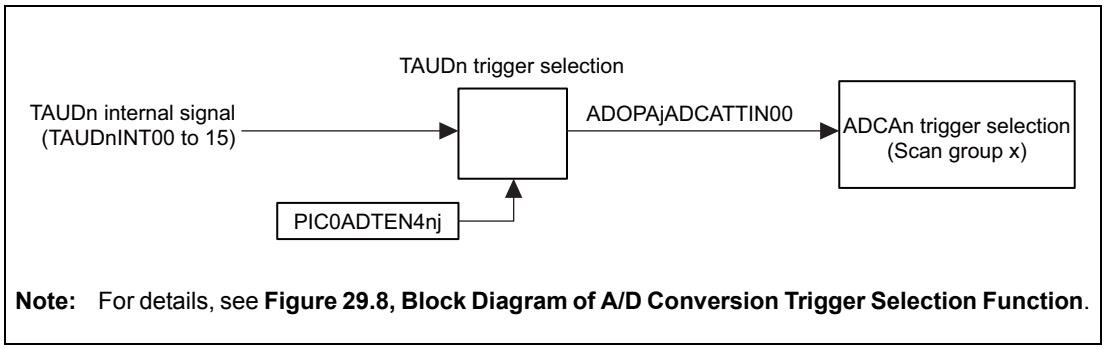
Restart

# 29.7 ADCA Trigger Selection Function

## 29.7.1 Functional Overview

This function generates ADCA hardware trigger signals by using TAUDn channel output.

## 29.7.2 Configuration




**Note:** For details, see **Figure 29.8, Block Diagram of A/D Conversion Trigger Selection Function**.

**Figure 29.12 Block Diagram of ADCA Trigger Selection Function**

## 29.7.3 Registers

### 29.7.3.1 PIC0ADTEN4nj — A/D Conversion Trigger Output Control Register 4nj (n = 0, j = 0 to 2)

This register selects an ADCA0 start trigger source from TAUDn channel m. (m = 0 to 15)

**Access:** This register can be read or written in 16-bit units.

**Address:** <PIC0_base> + 90H + 4 × j

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PIC0ADTEN4nj15 | PIC0ADTEN4nj14 | PIC0ADTEN4nj13 | PIC0ADTEN4nj12 | PIC0ADTEN4nj11 | PIC0ADTEN4nj10 | PIC0ADTEN4nj09 | PIC0ADTEN4nj08 | PIC0ADTEN4nj07 | PIC0ADTEN4nj06 | PIC0ADTEN4nj05 | PIC0ADTEN4nj04 | PIC0ADTEN4nj03 | PIC0ADTEN4nj02 | PIC0ADTEN4nj01 | PIC0ADTEN4nj00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.25 PIC0ADTEN4nj Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | PIC0ADTEN4nj 15 to PIC0ADTEN4nj 00 | Sets a trigger source of CHm (m = 0 to 15) in the TAUDn timer.<br>0: A/D trigger source of CHm in the TAUDn timer is disabled.<br>1: A/D trigger source of CHm in the TAUDn timer is enabled. |

### 29.7.4 Example of Operation

(1) Initial setting: Set the function of each channel of the TAUD0 to be used.

(2) Setting of the A/D conversion trigger output control register 4nj (PIC0ADTEN4nj):
Setting the bits of A/D conversion trigger output control register 4nj ((PIC0ADTEN4nj) to 1 enables selection of an interrupt request signal from each channel of the TAUD0 as the trigger of the A/D conversion scan group.

– Register setting should be performed when A/D conversion is stopped.

(3) Setting of the A/D conversion trigger selection control register (ADCA0SGTSELx):
Setting the bits corresponding to each trigger to 1 enables to use the signal generated by executing the logical OR of each trigger as the start trigger of the A/D conversion scan group.

– Register setting should be performed when the A/D conversion is stopped.

(4) Enabling of TAUD0 timer operation
Each channel of the TAUD0 timer set in (1) starts.

### 29.7.5 Setup Flow

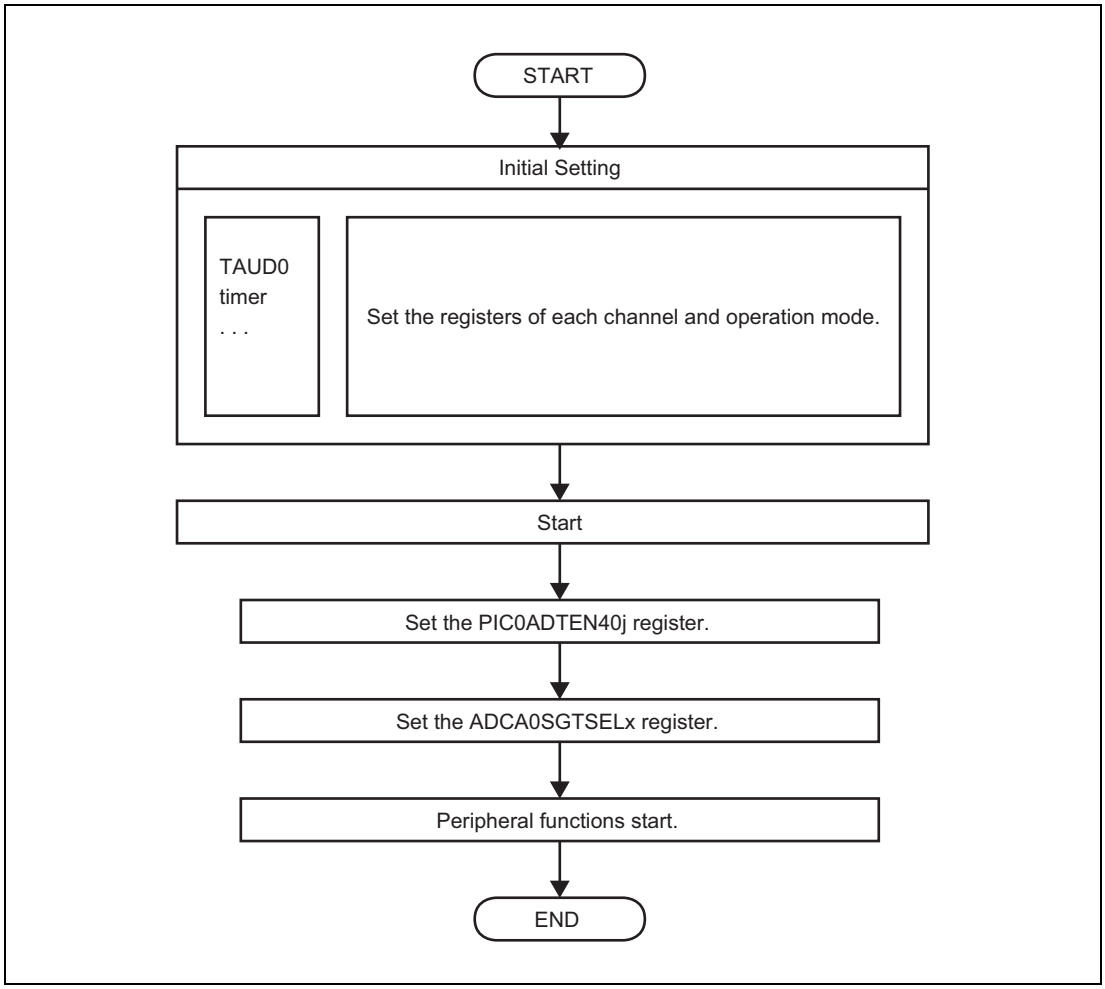



**Figure 29.13 Setup Flow (j = 0 to 2)**

# 29.8 Simultaneous Start Trigger Function

## 29.8.1 Functional Overview

The timers (TAUD0, TAUJ1, ENCA0) can be simultaneously started in any combination.

## 29.8.2 Configuration

### (1) Configuration

Table 29.26 Configuration of Simultaneous Start Trigger Function

| Configuration/Timer Function | Timer |
|---|---|
| Configuration of Timer | TAUD0, TAUJ1, ENCA0 |

### (2) Block Diagram

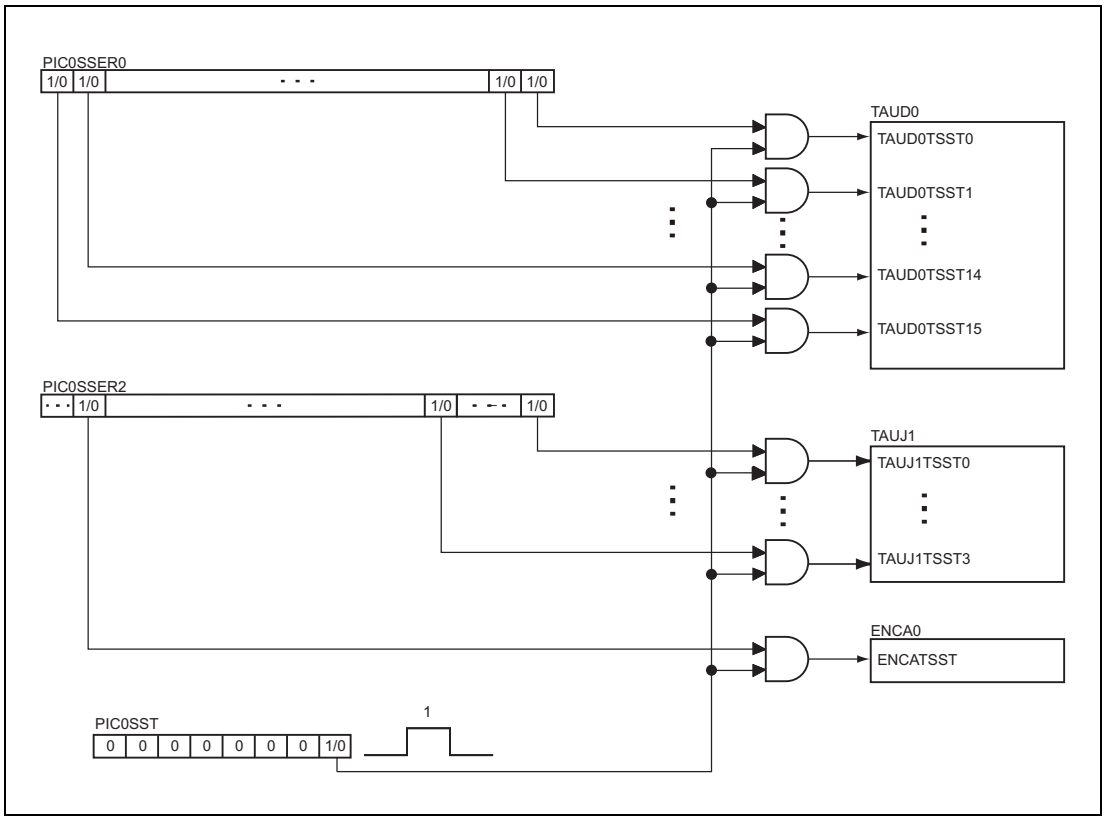



Figure 29.14 Block Diagram of Simultaneous Start Trigger

## 29.8.3 Registers

### 29.8.3.1 PIC0SSER0 — Simultaneous Start Control Register 0

The PIC0SSER0 register enables a start trigger for each channel of the TAUD0.

**Access:** This register can be read or written in 16-bit units

**Address:** <PIC0_base> + $10_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PIC0SSER015 | PIC0SSER014 | PIC0SSER013 | PIC0SSER012 | PIC0SSER011 | PIC0SSER010 | PIC0SSER009 | PIC0SSER008 | PIC0SSER007 | PIC0SSER006 | PIC0SSER005 | PIC0SSER004 | PIC0SSER003 | PIC0SSER002 | PIC0SSER001 | PIC0SSER000 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.27 PIC0SSER0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | PIC0SSER015 to PIC0SSER000 | Enable a simultaneous start trigger for the CHm in the TAUD0.<br>0: Simultaneous start trigger is disabled.<br>1: Simultaneous start trigger is enabled. |

### 29.8.3.2 PIC0SSER2 — Simultaneous Start Control Register 2

The PIC0SSER2 register enables a start trigger for ENCA0 and TAUJ1.

**Access:** This register can be read or written in 16-bit units

**Address:** <PIC0_base> + $18_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | PIC0SSER214(ENCA0) | — | — | — | — | — | — | — | — | — | — | PIC0SSER203 | PIC0SSER202 | PIC0SSER201 | PIC0SSER200 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 29.28 PIC0SSER2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 14 | PIC0SSER214 | Enables a simultaneous start trigger for the ENCA0 timer.<br>0: Simultaneous start trigger is disabled.<br>1: Simultaneous start trigger is enabled. |
| 13 to 4 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 3 to 0 | PIC0SSER203 to PIC0SSER200 | Set a simultaneous start trigger for the CHm in the TAUJ1 timer.<br>0: Simultaneous start trigger is disabled.<br>1: Simultaneous start trigger is enabled. |

### 29.8.3.3 PIC0SST — Simultaneous Start Trigger Control Register

**Access:** This register is a write-only register that can be written in 8-bit units.

**Address:** <PIC0_base> + $04_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | SYNCTRG |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 29.29 PIC0SST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | SYNCTRG | Generates a start trigger for the timer whose simultaneous start is enabled.<br>When read, this bit is always read as 0.<br>0: Disabled.<br>1: Simultaneous start trigger is generated (the pulse in the width of 1PCLK is output). |

### 29.8.4 Example of Operation

(1) Operation example of timer configuration:
The timers that operates in operation mode to be selected can be simultaneously started in any combinations.

(2) Simultaneous start enable:
Setting the relevant bits in the PIC0SSER0 and PIC0SSER2 registers of the target timers to be simultaneously started to 1 enables these timers to simultaneously start.

(3) Start trigger output:
Writing 1 to the SYNCTRG bit in the PIC0SST register enables the target timers set in (2) to simultaneously start.

(4) Repeating (2) and (3) for the channels that have not started yet enables the different target timers to simultaneously start in multiple batches.

### 29.8.5 Setup Flow

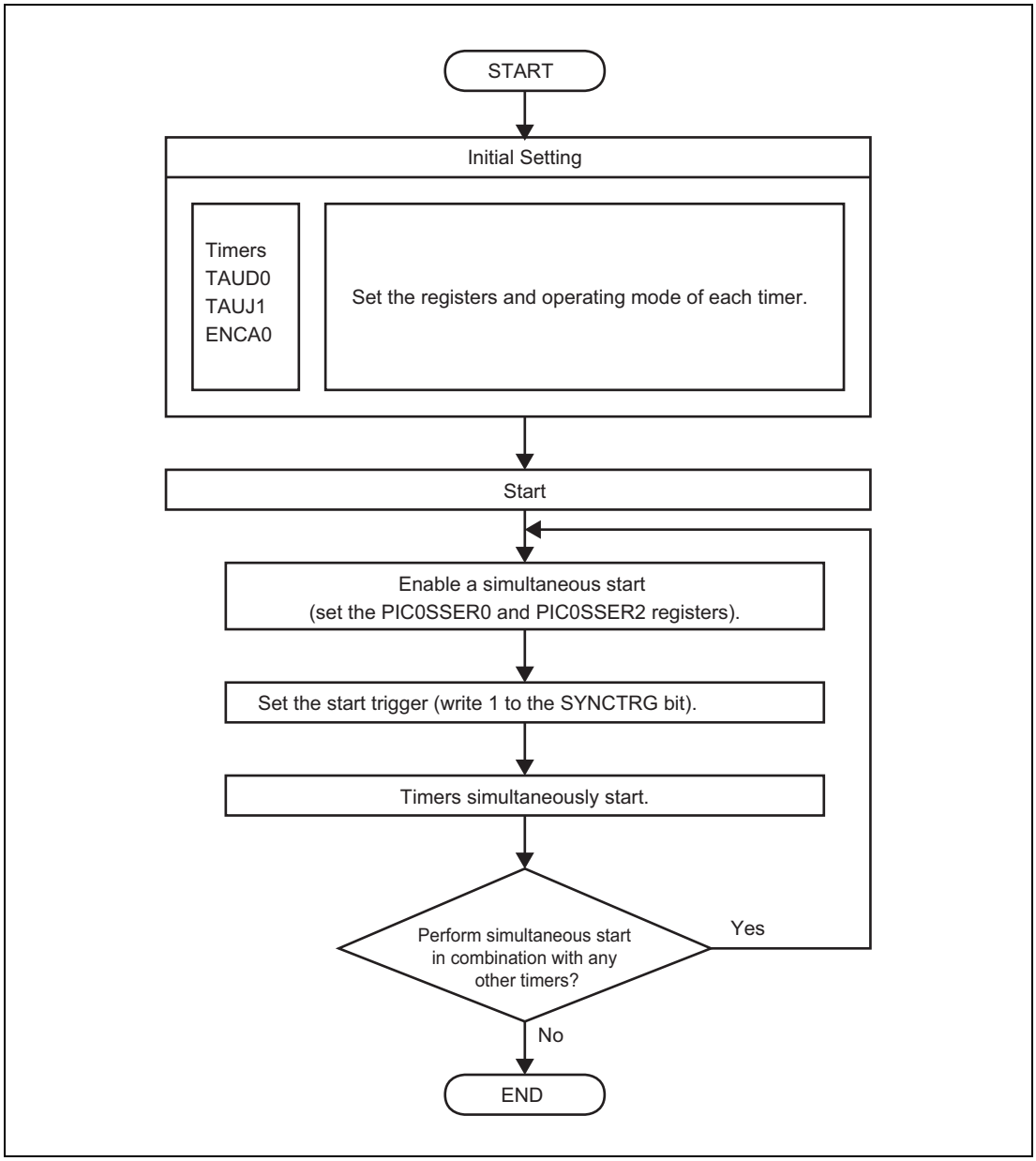



**Figure 29.15 Setup Flow**

# 29.9 Trigger & Pulse Width Measuring Function

## 29.9.1 Functional Overview

This function allows measurement of trigger periods by inputting the trigger signal output from ENCA0 to TAUJ1 and TAUD0.

## 29.9.2 Configuration

### (1) Configuration

**Table 29.30 Configuration of Trigger & Pulse Width Measuring Function**

| Configuration/Timer Function | Timer |
|---|---|
| Configuration of Timer | ENCA0, TAUD0, TAUJ1 |

**Table 29.31 Setting Functions of TAUJ1/TAUD0 Channels**

| TAU | Channels | Functions Name | M/S[*1] | Target Trigger of Pulse Width Measurement |
|---|---|---|---|---|
| TAUJ1 | 00 | TINm input pulse interval measurement function | S | ENCAT0IEC[*2] |
| | 01 | TINm input pulse interval measurement function | S | ENCAT0IEC[*2] |
| TAUD0 | 00 | TINm input pulse interval measurement function | S | ENCAT0EQ0, ENCAT0EQ1 |
| | 01 | TINm input pulse interval measurement function | S | ENCAT0EQ1 |
| | 02 | TINm input pulse interval measurement function | S | ENCAT0EQ0 |

Note 1. M: Master channel, S: Slave channel

Note 2. Read ENCAT0IEC as ENCATIEC (encoder clear interrupt) in **Table 28.5, Interrupt Requests**.

### (2) Block Diagram

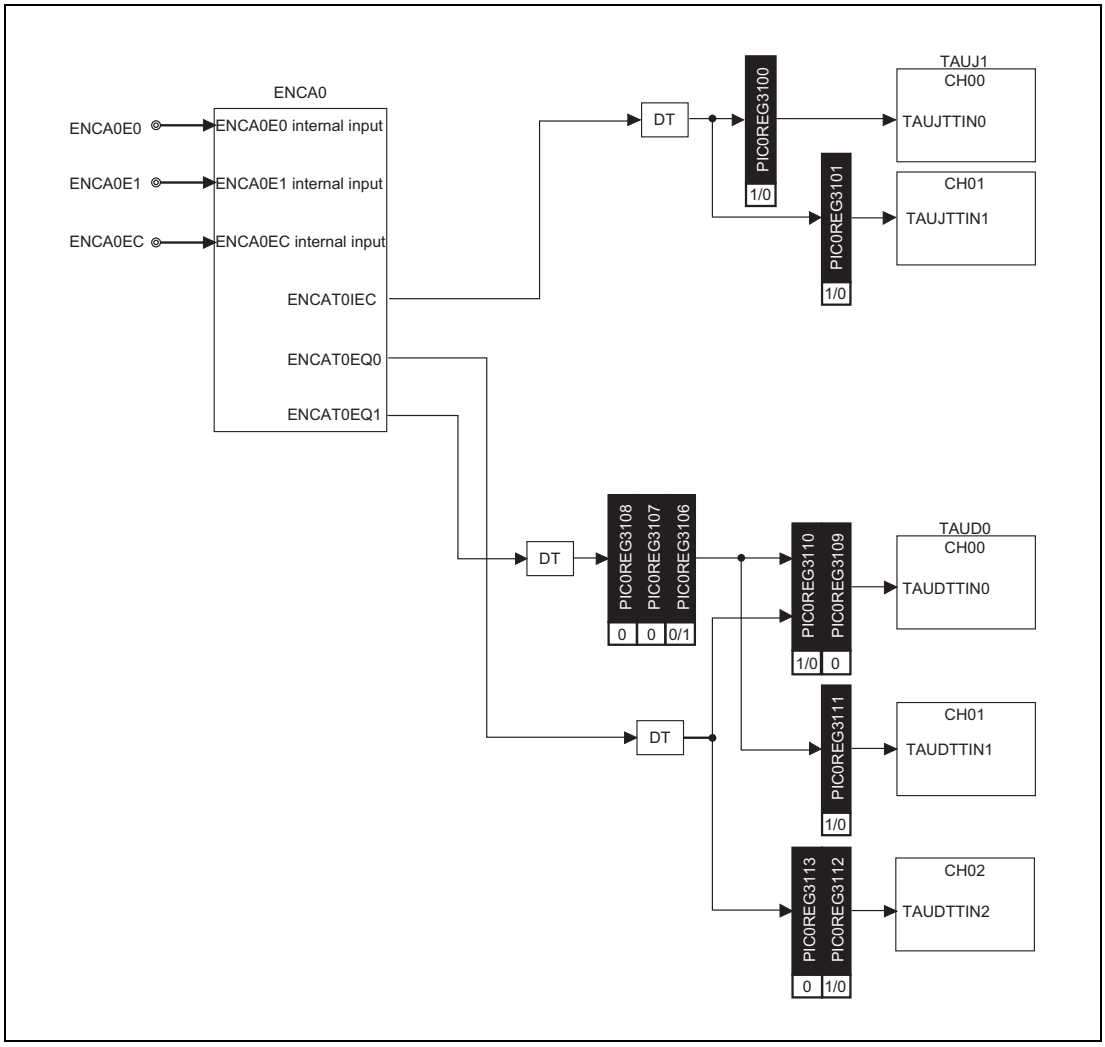



**Figure 29.16 Block Diagram of Trigger & Pulse Width Measuring Function**

## 29.9.3 Registers

### 29.9.3.1 PIC0REG31 — Timer I/O Control Register 31

**Access:** This register can be read or written in 32-bit units.

**Address:** <PIC0_base> + $EC_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | PIC0REG3113 | PIC0REG3112 | PIC0REG3111 | PIC0REG3110 | PIC0REG3109 | PIC0REG3108 | PIC0REG3107 | PIC0REG3106 | — | — | — | — | PIC0REG3101 | PIC0REG3100 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W |

**Table 29.32 PIC0REG31 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 14 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 13, 12 | PIC0REG3113 to PIC0REG3112 | Select a TIN input signal to CH2 of TAUD0. |

| PIC0REG3113 | PIC0REG3112 | Input Signal |
|---|---|---|
| 0 | 0 | CH2 of TAUD0 is not used for trigger width measurement. |
| 0 | 1 | DT output signal of ENCAT0EQ0 |
| Other than above | | Setting prohibited |

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 | PIC0REG3111 | Select a TIN input signal to CH1 of TAUD0.<br>0: CH1 of TAUD0 is not used for trigger width measurement.<br>1: Signal selected in PIC0REG3106 to PIC0REG3108 (when measuring the ENCATEQ1 signal) |
| 10, 9 | PIC0REG3110 to PIC0REG3109 | Select a TIN input signal to CH0 of TAUD0. |

| PIC0REG3110 | PIC0REG3109 | Input Signal |
|---|---|---|
| 0 | 0 | Signal selected in PIC0REG3106 to PIC0REG3108 |
| 1 | 0 | DT output signal of ENCAT0EQ0 |
| Other than above | | Setting prohibited |

| Bit Position | Bit Name | Function |
|---|---|---|
| 8 to 6 | PIC0REG3108 to PIC0REG3106 | Select a TIN input signal to CH0 and CH1 of TAUD0. |

| PIC0REG3108 | PIC0REG3107 | PIC0REG3106 | Input Signal |
|---|---|---|---|
| 0 | 0 | 0 | CH0 of TAUD0 is not used for trigger width measurement. |
| 0 | 0 | 1 | DT output signal of ENCAT0EQ1 |
| Other than above | | | Setting prohibited |

| Bit Position | Bit Name | Function |
|---|---|---|
| 5 to 2 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

Table 29.32 PIC0REG31 Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 1 | PIC0REG3101 | Selects a TIN input signal to CH1 of TAUJ1.<br>0: CH1 of TAUJ1 is not used for trigger width measurement.<br>1: DT output signal of ENCAT0IEC |
| 0 | PIC0REG3100 | Selects a TIN input signal to CH0 of TAUJ1.<br>0: CH0 of TAUJ1 is not used for trigger width measurement.<br>1: DT output signal of ENCAT0IEC |

### 29.9.4 Example of Operation

The trigger and pulse width measurement function is achieved by combining the ENCA0 trigger signals (ENCAT0IEC, ENCAT0EQ0, ENCAT0EQ1) and the following functions of TAUD0 and TAUJ1.

- TAUDTTINm input pulse interval measurement function (TAUD0)
- TAUJTTINm input pulse interval measurement function (TAUJ1)

Also, the following function of PIC is used to convert the trigger signal input to TINm into a level-sensitive toggle signal.

- DT circuit

The trigger and pulse width measurement function implements measurement of the ENCA0 output trigger signal interval using the TAUDTTINm input pulse interval measurement function of TAUD0 and the TAUJTTINm input pulse interval measurement function of TAUJ1.

#### (1) TAUDTTINm input pulse interval measurement function, TAUJTTINm input pulse interval measurement function

When the valid TINm edge of TAUD0 or TAUJ1 is detected, the CNTm value is captured into CDRm and CNTm is cleared.

**CAUTION**

**Set both edges (rising and falling edges) of TINm to be detected as valid (TAUD0CMURm.TAUD0TIS[1:0] = $10_B$, TAUJ1CMURm.TAUJ1TIS[1:0] = $10_B$) for this function.**

For details of the TAUD and TAUJ functions, see the corresponding sections.

#### (2) DT circuit

The DT circuit is used to convert the trigger signal output from ENCA0 into a level-sensitive toggle signal.

As shown in **Figure 29.17, Operation of DT Circuit**, the output signal is toggled upon each input trigger signal generation.

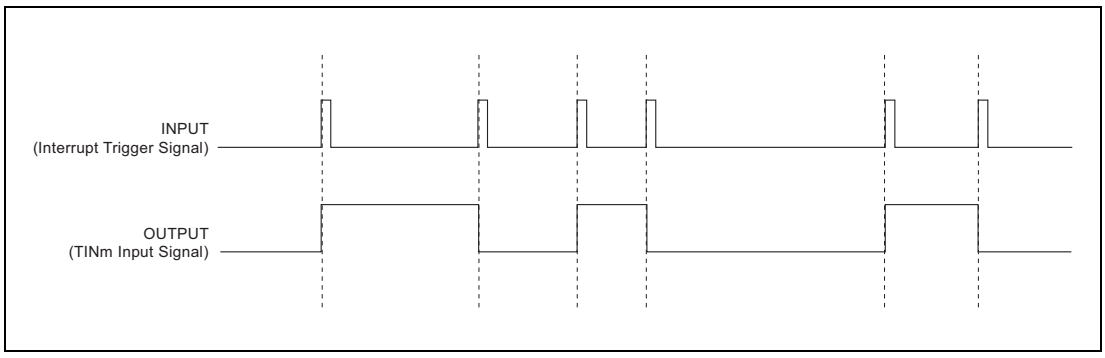



**Figure 29.17 Operation of DT Circuit**

PIC provides input signal conversion and signal connection to TAUD0 and TAUJ1 to measure the generation interval of trigger signals from ENCA0.

The timing chart of the trigger and pulse width measurement function is shown below.

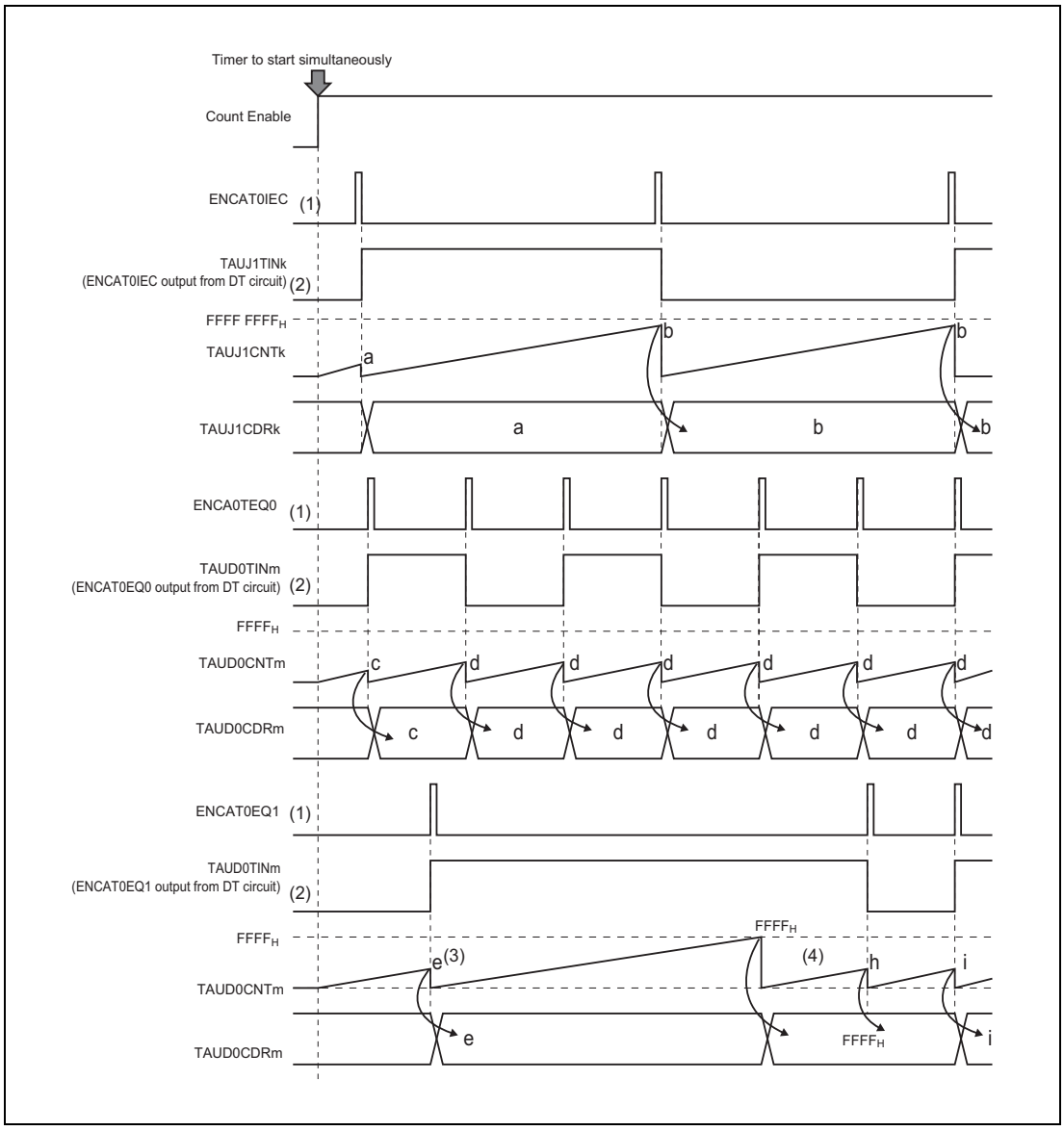



**Figure 29.18 Operation Example of the Function of Trigger and Pulse Width Measurement (m = 0 to 2, k = 0, 1)**

(1) The following signals are output from ENCA0 as triggers:

- ENCAT0IEC (interrupt trigger signal output when timer counter value is cleared by ENCA0EC input)
- ENCAT0EQ0 (trigger signal output according to timing of a match of timer counter value and value of compare register 0)
- ENCAT0EQ1 (trigger signal output according to timing of a match of timer counter value and value of compare register 1)

(2) The trigger signal output from ENCA0 is converted to a level-sensitive toggle signal by the DT circuit and is output to TINm of TAUD0 and TAUJ1.

(3) By setting both TINm edges (rising and falling edges) of TAUD0 and TAUJ1 as valid, the CNTm value is captured into CDRm at the TINm toggle timing and cleared to $0000_H$. This operation is repeated.

The first captured value (shown as “a” in the figure) from the start of operation indicates the interval from the start of TAUJ operation until trigger input.

(4) When an overflow occurs, the count value $FFFF_H$ (FFFF $FFFF_H$ for TAUJ) is captured but the count value is not captured at the first trigger after the overflow.

With the above operation, the trigger generation interval can be measured.

The following table shows the combinations of the trigger signals and measurement timers, and the bit settings of the pertinent PIC registers for setting the signal paths and the I/O selection registers. Appropriately set these bits according to the trigger signal to be measured and the measurement timer to be used.

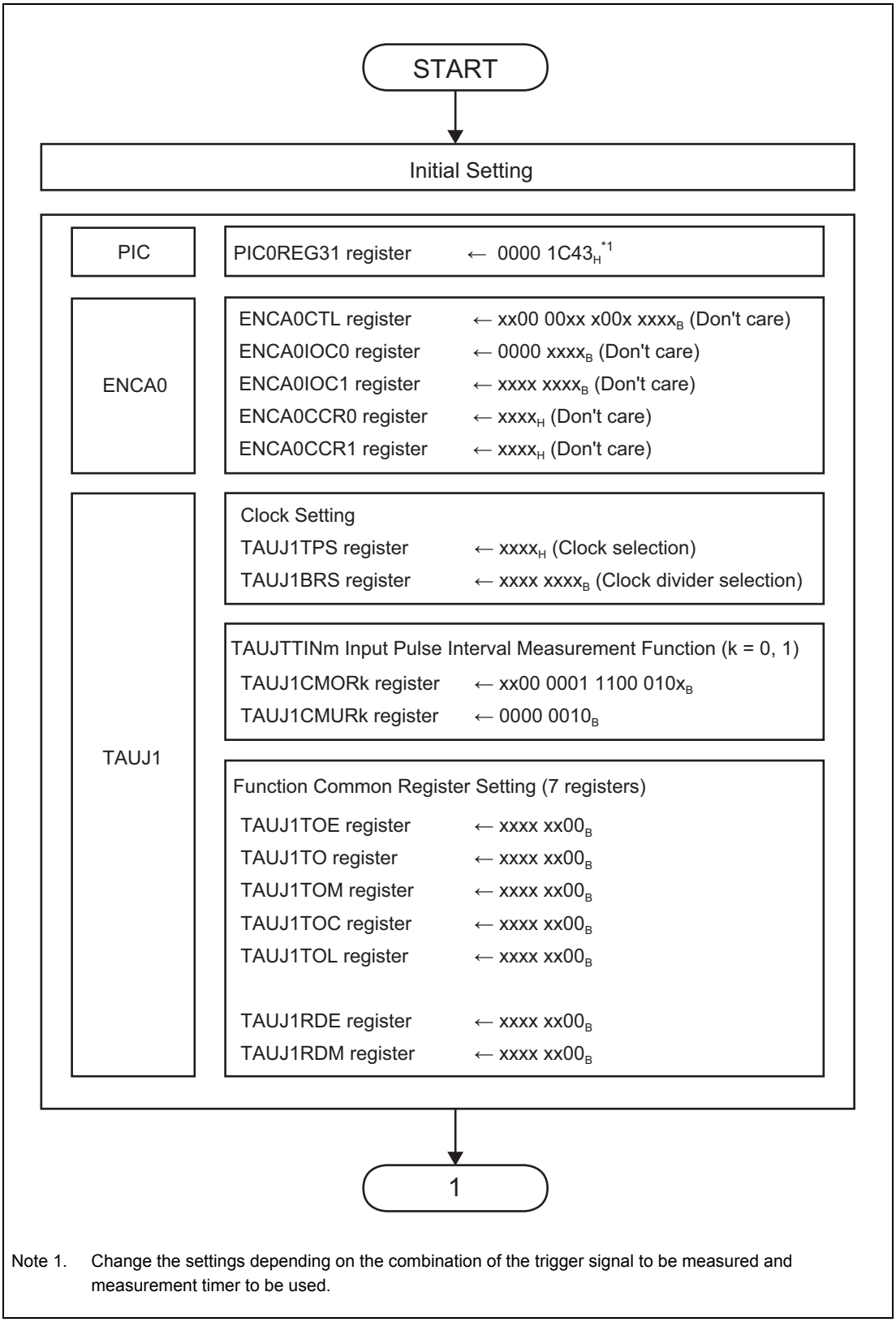
**Table 29.33 Combinations of Trigger Signals and Measurement Timers**

| Interrupt Trigger Signal | Measurement Timer | PIC Register Bit Setting | |
|---|---|---|---|
| ENCAT0IEC | TAUJ1 CH0 | PIC0REG3100 = 1 | |
| | TAUJ1 CH1 | PIC0REG3101 = 1 | |
| ENCAT0EQ0 | TAUD0 CH0 | PIC0REG3109 = 0<br>PIC0REG3110 = 1 | |
| | TAUD0 CH2 | PIC0REG3112 = 1<br>PIC0REG3113 = 0 | |
| ENCAT0EQ1 | TAUD0 CH0 | PIC0REG3106 = 1<br>PIC0REG3107 = 0<br>PIC0REG3108 = 0 | PIC0REG3109 = 0<br>PIC0REG3110 = 0 |
| | TAUD0 CH1 | | PIC0REG3111 = 1 |

## 29.9.5 Setup Flow

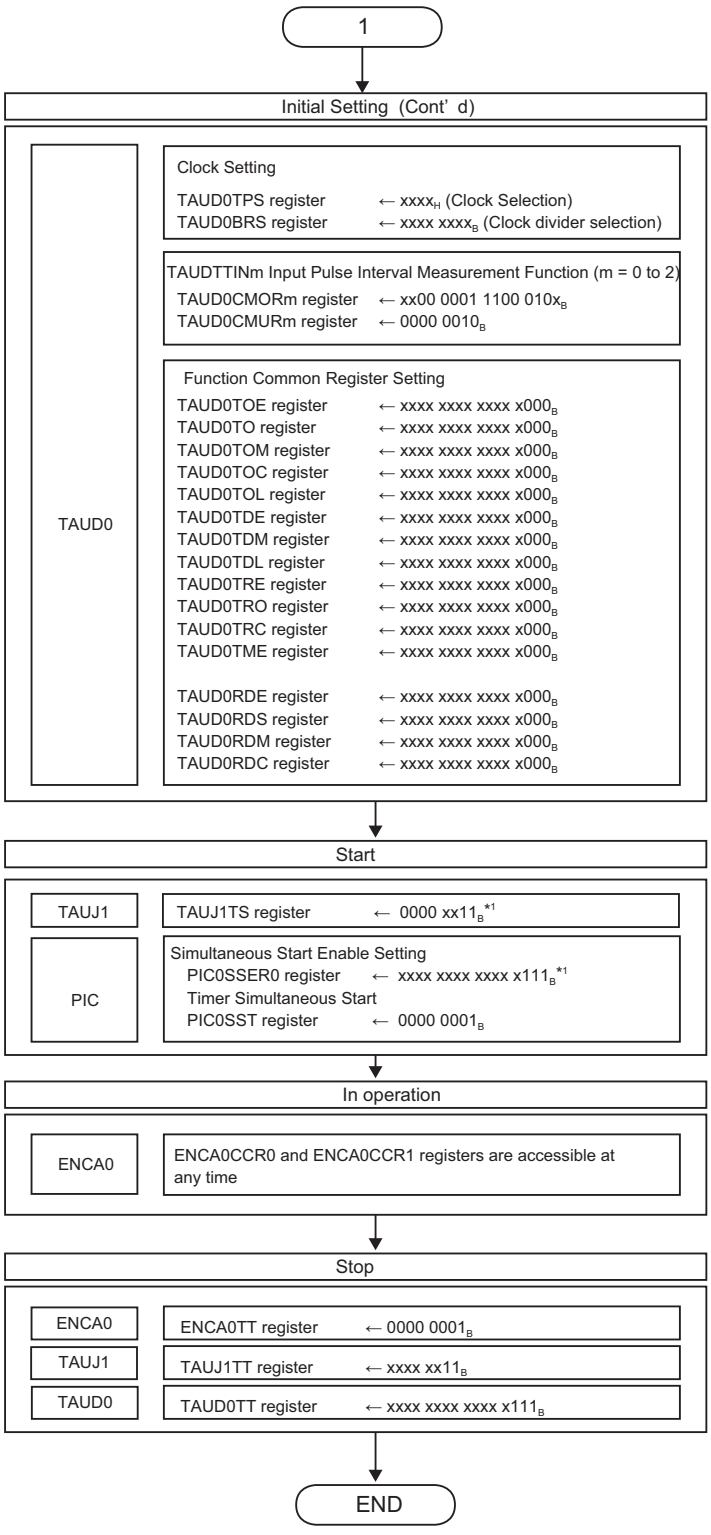
The setup flow in this section shows the general setup flow to measure the pulse interval, which applies to all the following combinations. For the combinations of the trigger signals and measurement timers, see **Table 29.33, Combinations of Trigger Signals and Measurement Timers**.

| Encoder Timer | Trigger Signal | Measurement Timer |
|---|---|---|
| ENCA0 | ENCAT0IEC | TAUJ1 CH0, TAUJ1 CH1 |
| | ENCAT0EQ0 | TAUD0 CH0, TAUD0 CH2 |
| | ENCAT0EQ1 | TAUD0 CH0, TAUD0 CH1 |




Note 1. Change the settings depending on the combination of the trigger signal to be measured and measurement timer to be used.

**Figure 29.19 Setup Flow**




Note 1. Change the settings depending on the combination of the trigger signal to be measured and measurement timer to be used.

**Figure 29.20 Setup Flow (Cont'd)**

### 29.9.6 Setting Examples for Operation Functions

This section provides example settings for each register.

The setup example shown in this section describes how to set up measurement of the pulse interval for all the combinations below. For the combinations of the trigger signals and measurement timers, see **Table 29.33, Combinations of Trigger Signals and Measurement Timers**.

| Encoder Timer | Trigger Signal | Measurement Timer |
|---|---|---|
| ENCA0 | ENCAT0IEC | TAUJ1 CH0, TAUJ1 CH1 |
| | ENCAT0EQ0 | TAUD0 CH0, TAUD0 CH2 |
| | ENCAT0EQ1 | TAUD0 CH0, TAUD0 CH1 |

**Table 29.34 ENCA0 Setting**

| Register | Bit Position | Bit Name | Setting Value | Note |
|---|---|---|---|---|
| ENCA0CTL | 15 | ENCA0CME | Don't care | Enables or disables compare match interrupt detection mask |
| | 14 | ENCA0MCS | Don't care | Selects a cancelation trigger for compare match interrupt detection mask |
| | 13 to 10 | | 0 | Fixed to 0 |
| | 9 | ENCA0CRM1 | Don't care | Selects the ENCA0CCR1 register function |
| | 8 | ENCA0CRM0 | Don't care | Selects the ENCA0CCR0 register function |
| | 7 | ENCA0CTS | Don't care | Selects trigger for capture operation of ENCA0CCR1. |
| | 6, 5 | | 0 | Fixed to 0 |
| | 4 | ENCA0LDE | Don't care | Enables or disables reload operation when underflow is generated |
| | 3 | ENCA0ECM1 | Don't care | Enables or disables clearing of the counter on compare match of ENCA0CCR1 |
| | 2 | ENCA0ECM0 | Don't care | Enables or disables clearing of the counter on compare match of ENCA0CCR0 |
| | 1, 0 | ENCA0UDS[1:0] | Don't care | Selects the counter up/down control by ENCA0E0 and ENCA0E1 |
| ENCA0IOC0 | 7 to 4 | | 0 | Fixed to 0 |
| | 3, 2 | ENCA0TIS[3:2] | Don't care | Selects the valid edge for capture trigger 1 (ENCA0I1) |
| | 1, 0 | ENCA0TIS[1:0] | Don't care | Selects the valid edge for capture trigger 0 (ENCA0I0) |
| ENCA0IOC1 | 7 | ENCA0SCE | Don't care | Enables encoder special-clear |
| | 6 | ENCA0ZCL | Don't care | Selects the clear level of Z phase for a encoder special-clear |
| | 5 | ENCA0BCL | Don't care | Selects the clear level of B phase for a encoder special-clear |
| | 4 | ENCA0ACL | Don't care | Selects the clear level of A phase for a encoder special-clear |
| | 3, 2 | ENCA0ECS[1:0] | Don't care | Selects encoder clear input (Z phase) edge |
| | 1, 0 | ENCA0EIS[1:0] | Don't care | Selects encoder input (A or B phase) edge |

**Table 29.35 TAUJ1 Setting (k = 0, 1)**
**TAUJ1 (TAUJTTINm Input Pulse Interval Measurement Function)**

| Register | Bit Position | Bit Name | Setting Value | Note |
|---|---|---|---|---|
| TAUJ1CMORk | 15,14 | TAUJ1CKS[1:0] | Don't care | Operation Clock Setting |
| | 13,12 | TAUJ1CCS[1:0] | 00 | |
| | 11 | TAUJ1MAS | 0 | |
| | 10, 9, 8 | TAUJ1STS[2:0] | 001 | |
| | 7, 6 | TAUJ1COS[1:0] | 11 | |
| | 5 | | 0 | Fixed to 0 |
| | 4, 3, 2, 1 | TAUJ1MD[4:1] | 0010 | |
| | 0 | TAUJ1MD0 | Don't care | |
| TAUJ1CMURk | 1, 0 | TAUJ1TIS[1:0] | 10 | |

**NOTE**

When TAUJ1CMORk is used for the TAUDTTINm input pulse interval measurement function, the TAUJ1CKS[1:0] (operating clock selection) and TAUJ1MD0 (INTm output control at the start of counting) bits can be set arbitrarily.

Although the TAUJ1COS[1:0] (overflow mode selection) bits can also be set arbitrarily, these bits should be fixed values as specified above for this function.

Other control bits have fixed values as specified above. For details, see **Section 26, Timer Array Unit J (TAUJ)**.

For TAUJ common registers (TAUJ1TOE, TAUJ1TO, TAUJ1TOM, TAUJ1TOC, TAUJ1TOL,TAUJ1RDE, and TAUJ1RDM), only set the bits corresponding to the used channels to 0.

**Table 29.36 TAUD0 Setting (m = 0 to 2)**
**TAUD0 (TAUDTTINm Input Pulse Interval Measurement Function)**

| Register | Bit Position | Bit Name | Setting Value | Note |
|---|---|---|---|---|
| TAUD0CMORm | 15, 14 | TAUD0CKS[1:0] | Don't care | Operation Clock Setting |
| | 13, 12 | TAUD0CCS[1:0] | 00 | |
| | 11 | TAUD0MAS | 0 | |
| | 10 to 8 | TAUD0STS[2:0] | 001 | |
| | 7, 6 | TAUD0COS[1:0] | 11 | |
| | 5 | | 0 | Fixed to 0 |
| | 4 to 1 | TAUD0MD[4:1] | 0010 | |
| | 0 | TAUD0MD0 | Don't care | |
| TAUD0CMURm | 1, 0 | TAUD0TIS[1:0] | 10 | |

**NOTE**

When TAUD0CMORm is used for the TAUDTTINm input pulse interval measurement function, the TAUD0CKS[1:0] (operating clock selection) and TAUD0MD0 (INTm output control at the start of counting) bits can be set arbitrarily.

Although the TAUD0COS[1:0] (overflow mode selection) bits can also be set arbitrarily, these bits should be fixed values as specified above for this function.

Other control bits have fixed values as specified above. For details, see **Section 25, Timer Array Unit D (TAUD)**

For TAUD common registers (TAUD0TOE, TAUD0TO, TAUD0TOM, TAUD0TOC,TAUD0TOL, TAUD0TDE, TAUD0TDM, TAUD0TDL, TAUD0TRE, TAUD0TRO,TAUD0TRC, TAUD0TME, TAUD0RDE, TAUD0RDS, TAUD0RDM, and TAUD0RDC), only set the bits corresponding to the used channels to 0.

**Table 29.37 PIC Setting**

| Register | Bit Position | Bit Name | Setting Value | Note |
|---|---|---|---|---|
| PIC0REG31 | 13, 12 | PIC0REG3113<br>PIC0REG3112 | 0<br>1 | Selects the DT output signal from ENCAT0EQ0 as TAUD0TTIN2 input signal |
| | 11 | PIC0REG3111 | 1 | Selects the signal selected with PIC0REG3106 to PIC0REG3108 (DT output signal from ENCAT0EQ1) as TAUD0TTIN1 input signal |
| | 10, 9 | PIC0REG3110<br>PIC0REG3109 | 1<br>0 | Selects the DT output signal from ENCAT0EQ0 as TAUD0TTIN0 input signal |
| | 8 to 6 | PIC0REG3108<br>PIC0REG3107<br>PIC0REG3106 | 0<br>0<br>1 | Selects the DT output signal from ENCAT0EQ1 as TAUD0TTIN1 or TAUD0TTIN0 input signal |
| | 1 | PIC0REG3101 | 1 | Selects the DT output signal from ENCAT0IEC as TAUJ1TTIN1 input signal |
| | 0 | PIC0REG3100 | 1 | Selects the DT output signal from ENCAT0IEC as TAUJ1TTIN0 input signal |

# 29.10 A/D Trigger Encoder Capture Function

## 29.10.1 Functional Overview

The value of the encoder counter synchronized with A/D conversion can be obtained by using an A/D conversion trigger signal as a capture signal of ENCA0.

## 29.10.2 Configuration

### (1) Configuration

Table 29.38 Configuration of A/D Trigger Encoder Capture Function

| A/D Converter | Encoder Timer |
|---|---|
| ADCA0 | ENCA0 |

### (2) Block Diagram

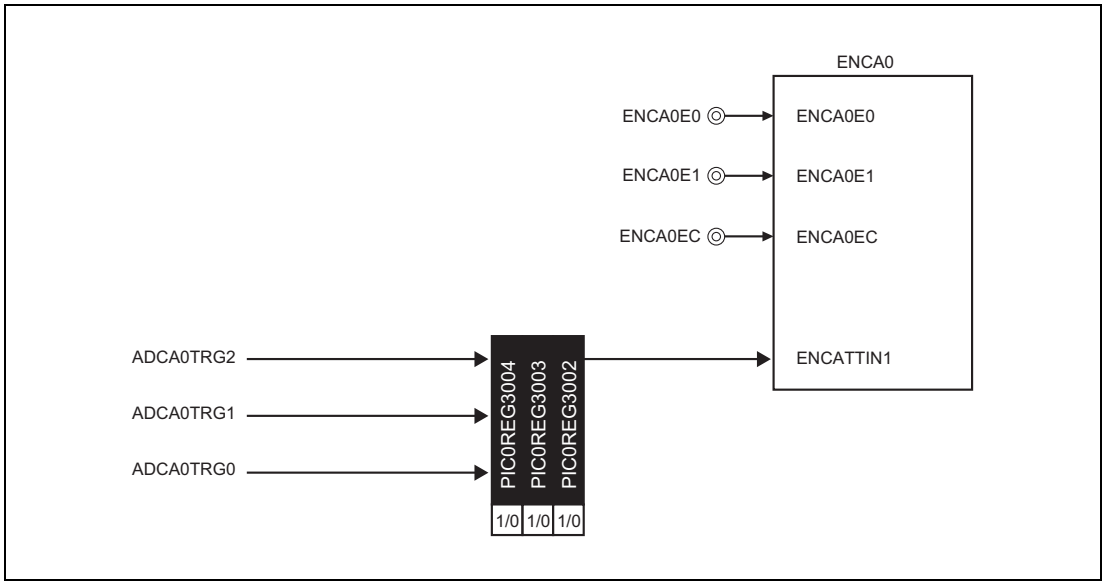



Figure 29.21 Block Diagram of A/D Trigger Encoder Capture Function

**CAUTIONS**

1. **It takes ENCA0 one additional clock cycle to capture the input signal of the ADCA0TRG0, ADCA0TRG1, and ADCA0TRG2 pins compared with the number of clock cycles it takes to capture the input signal of the ENCA0TIN1 pin when using CKSCLK_IPERI1, and three additional clock cycles when using CKSCLK_AADCA. Be sure to take this into account when configuring your system.**
2. **Configure the edge detection function by using the edge detection function registers of the digital noise filter, which are FCLA0CTL0_ADC0, FCLA0CTL1_ADC0, and FCLA0CTL2_ADC0 (see Section 2.12.1.4, Input Pins that Incorporate Digital Filter Type D, for details), and specify "rising edge" for edge detection of the ENCA0TIN1 capture trigger input of ENCA0 (ENCA0IOC0.ENCA0TIS [3:2] = $01_B$). Do not set ENCA0IOC0.ENCA0TIS [3:2] to $10_B$ (falling edge) or $11_B$ (both edges).**

## 29.10.3 Registers

### 29.10.3.1 PIC0REG30 — Timer I/O Control Register 30

**Access:** This register can be read or written in 32-bit units.

**Address:** <PIC0_base> + $E8_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | PIC0REG3004 | PIC0REG3003 | PIC0REG3002 | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R | R |

**Table 29.39 PIC0REG30 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 4 to 2 | PIC0REG3004 to PIC0REG3002 | Selects an input signal to ENCATTIN1. |
| 1, 0 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |

| PIC0REG3004 | PIC0REG3003 | PIC0REG3002 | Input signal |
|---|---|---|---|
| 0 | 0 | 0 | Capture is not performed by an A/D trigger signal in ENCA0. |
| 0 | 1 | 0 | ADCA0TRG2 |
| 0 | 1 | 1 | ADCA0TRG1 |
| 1 | 0 | 0 | ADCA0TRG0 |
| Other than above | | | Setting prohibited |

### 29.10.4 Example of Operation

The A/D trigger encoder capture function is implemented by connecting A/D conversion trigger signal ADCAnTRGi (n = 0, i = 0 to 2) to ENCA0.

**CAUTION**

**When using this function, the ENCA0 interrupt signal ENCATINT1 should not be selected as the A/D converter trigger. If selected, the correct operation cannot be performed because the following loop occurs: ADCAnTRG1 generation → ENCA0 capture operation → ENCATINT1 generation by capture operation → ADCAnTRG1 generation.**

The following shows a timing chart of the A/D trigger encoder capture function using the ADCA0TRG1 as a trigger.

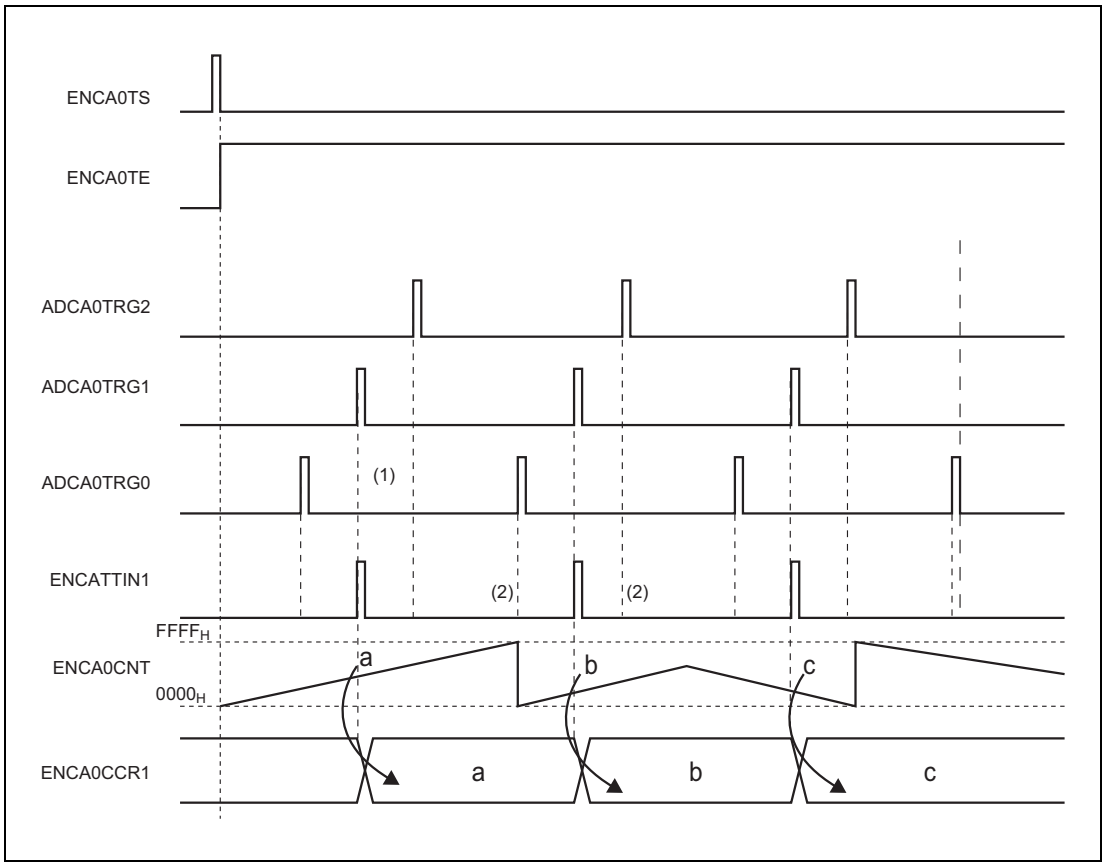


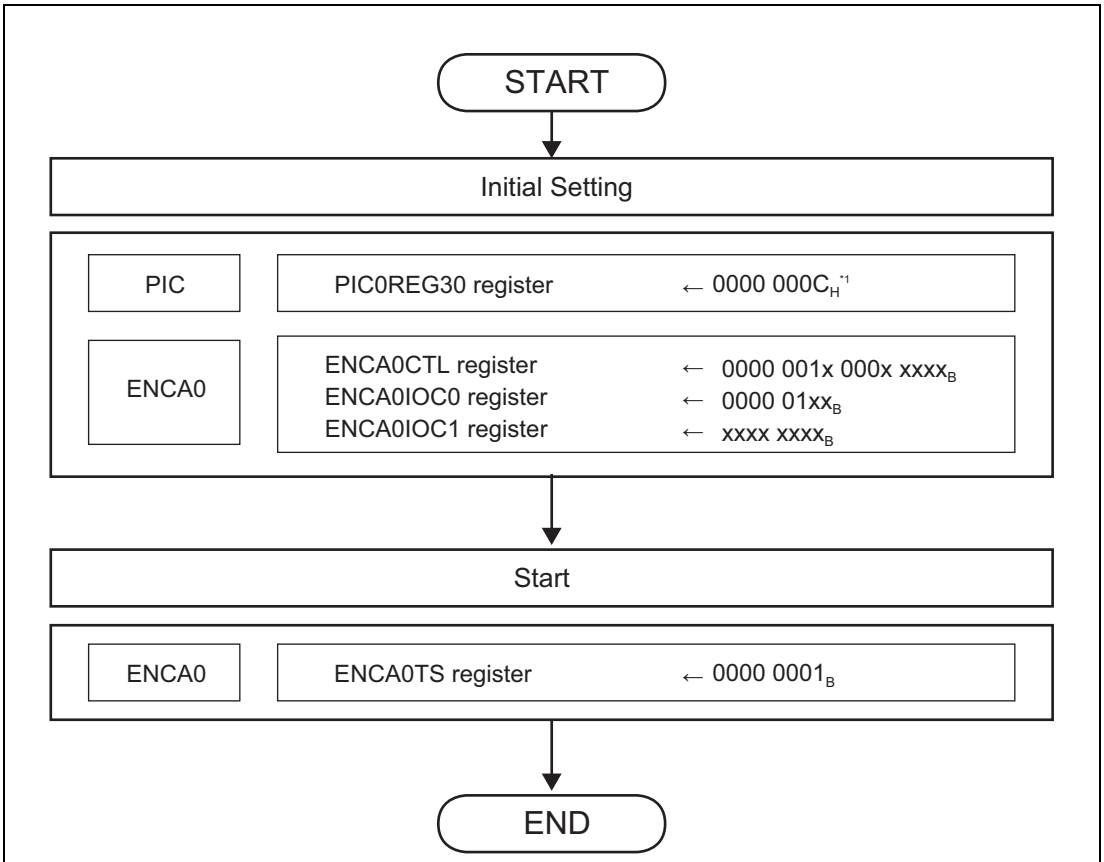

**Figure 29.22 Operation Example of Trigger Encoder Capture function**

(1) When ADCA0TRG1 is selected as ENCA0 capture trigger 1 signal ENCATTIN1, the valid ADCA0TRG1 is input to ENCA0 as the ENCATTIN1 signal and ENCA0 is captured.

(2) When a hardware trigger signal (ADCA0TRG0, ADCA0TRG2) other than ADCA0TRG1 is generated, the ENCATTIN1 signal is not generated and ENCA0 is not captured.

### 29.10.5 Setup Flow

The setup flow in this section shows the general setup flow to perform the capture operation of encoder timer ENCA0 based on the ADCA0TRG1 signal.




Note 1. Change the settings depending on the hardware trigger to perform the capture operation.

**Figure 29.23 Setup Flow**

### 29.10.6 Setting Examples for Operation Functions

This section provides example settings for each register.

The setup example shown in this section describes how to set up capture operation of encoder timer ENCA0 based on the ADCA0TRG1 signal. Change the settings depending on the hardware trigger to perform the capture operation.

**Table 29.40 ENCAn Setting**

| Register | Bit Position | Bit Name | Setting Value | Remark |
|---|---|---|---|---|
| ENCAnCTL | 15 | ENCAnCME | 0 | Disables compare match interrupt detection masking |
| | 14 | ENCAnMCS | 0 | Selects release trigger for compare match interrupt detection masking |
| | 13 to 10 | | 0 | Fixed to 0 |
| | 9 | ENCAnCRM1 | 1 | Sets the ENCAnCCR1 register for capture operation |
| | 8 | ENCAnCRM0 | Don't care | Selects the function of ENCAnCCR0 register |
| | 7 | ENCAnCTS | 0 | Selects ENCATTIN1 as trigger for capture operation |
| | 6, 5 | | 0 | Fixed to 0 |
| | 4 | ENCAnLDE | Don't care | Enables or disables reload operation when ENCAnCCR0 register underflow occurs |
| | 3 | ENCAnECM1 | Don't care | Enables or disables clearing of the counter on compare match of ENCAnCCR1 register |
| | 2 | ENCAnECM0 | Don't care | Enables or disables clearing of the counter on compare match of ENCAnCCR0 register |
| | 1,0 | ENCAnUDS[1:0] | Don't care | Select the counter up/down control by ENCAnE0 and ENCAnE1 |
| ENCAnIOC0 | 7 to 4 | | 0 | Fixed to 0 |
| | 3, 2 | ENCAnTIS[3:2] | 0[*1]<br>1[*1] | Select the valid edge of capture trigger 1 (ENCATTIN1) for the rising edge detection |
| | 1, 0 | ENCAnTIS[1:0] | Don't care | Select the valid edge of capture trigger 0 (ENCATTIN0) |
| ENCAnIOC1 | 7 | ENCAnSCE | Don't care | Enables encoder special-clear |
| | 6 | ENCAnZCL | Don't care | Selects the clear level (input level) of Z phase for encoder special-clear |
| | 5 | ENCAnBCL | Don't care | Selects the clear level (input level) of B phase for encoder special-clear |
| | 4 | ENCAnACL | Don't care | Selects the clear level (input level) of A phase for encoder special-clear |
| | 3, 2 | ENCAnECS[1:0] | Don't care | Select encoder clear input (Z phase) edge |
| | 1, 0 | ENCAnEIS[1:0] | Don't care | Select encoder input (A or B phase) edge |
| PIC0REG30 | 4 | PIC0REG3004 | Don't care | Selects ADCA0 trigger signal of ENCATTIN1 |
| | 3 | PIC0REG3003 | Don't care | |
| | 2 | PIC0REG3002 | Don't care | |

Note 1. Change the setting depending on the hardware trigger to perform the capture operation.

**NOTE**

Bits ENCA0CRM1 and ENCA0CTS in ENCA0CTL are fixed: ENCA0CRM1 = 1 (ENCA0CCR1 register function) and ENCA0CTS = 0 (trigger source of capture to the ENCA0CCR1 register). All the other bits can be set arbitrarily.

# 29.11 Three-Phase PWM Output with Dead Time

## 29.11.1 Functional Overview

This feature generates each of the set signals (active level timing signals) and clear signals (inactive level timing signals) once or less per cycle and then uses the results to output a three-phase PWM waveform with dead time.

For the PWM output feature of TAUD, only the clear timing used during each cycle is specified by specifying the duty value, but for the feature described here, the set timing can also be specified, which makes more flexible PWM output with dead time possible.

## 29.11.2 Configuration

The unit and channel configuration for this feature is shown below. (n = 0)

**Table 29.41 Configuration of Three-Phase PWM Output with Dead Time**

| Timer | Timer Motor Control Function |
|---|---|
| TAUD0 CH2, CH4 to CH15 (used channels fixed) | TAPA0 |

The signal names used in the descriptions below are abbreviations. The actual signal names corresponding to each abbreviation are as follows:

- INTm → INTTAUDnIm (TAUDn channel m interrupt)
- TINm → TAUDTTINm (TAUDn channel m input)
- TOUTm → TAUDTTOUTm (TAUDn channel m output)
- CDRm → TAUDnCDRm (TAUDn channel m data register)
- CNTm → TAUDnCNTm (TAUDn channel m counter register)

### (1) TAUDn configuration

Because CH10, CH12, and CH14 are only used for TOUTm, these channels can be used for features that do not use TOUTm (m = 10, 12, 14).

**Table 29.42 TAUDn Configuration (1/2)**

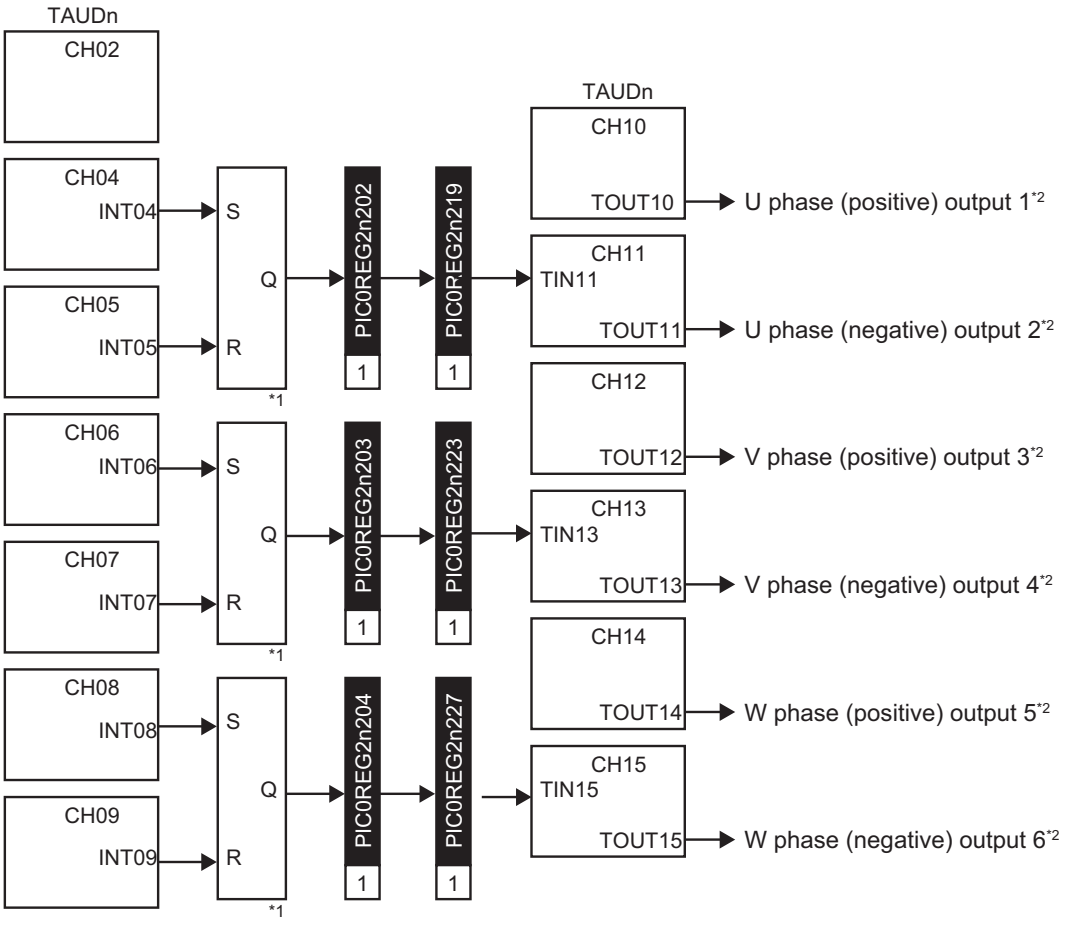
| CH | Function Name | M/S | CDR Setting | Description |
|---|---|---|---|---|
| 2 | PWM output (CH2 is the master channel for CH4 to CH9.) | M | Cycle | |
| 4 | | S | Duty (U phase signal setting) | |
| 5 | | S | Duty (U phase signal clearing) | |
| 6 | | S | Duty (V phase signal setting) | |
| 7 | | S | Duty (V phase signal clearing) | |
| 8 | | S | Duty (W phase signal setting) | |
| 9 | | S | Duty (W phase signal clearing) | |
| 10 | Any feature that does not use TOUT10 | S | | TOUT10: U phase output |
| 11 | One-phase PWM output | S | Dead time (U phase) | TOUT11: UB phase output |
| 12 | Any feature that does not use TOUT12 | S | | TOUT12: V phase output |

Table 29.42 TAUDn Configuration (2/2)

| CH | Function Name | M/S | CDR Setting | Description |
|---|---|---|---|---|
| 13 | One-phase PWM output | S | Dead time (V phase) | TOUT13: VB phase output |
| 14 | Any feature that does not use TOUT14 | S | | TOUT14: W phase output |
| 15 | One-phase PWM output | S | Dead time (W phase) | TOUT15: WB phase output |

**Note:** M: Master channel, S: Slave channel

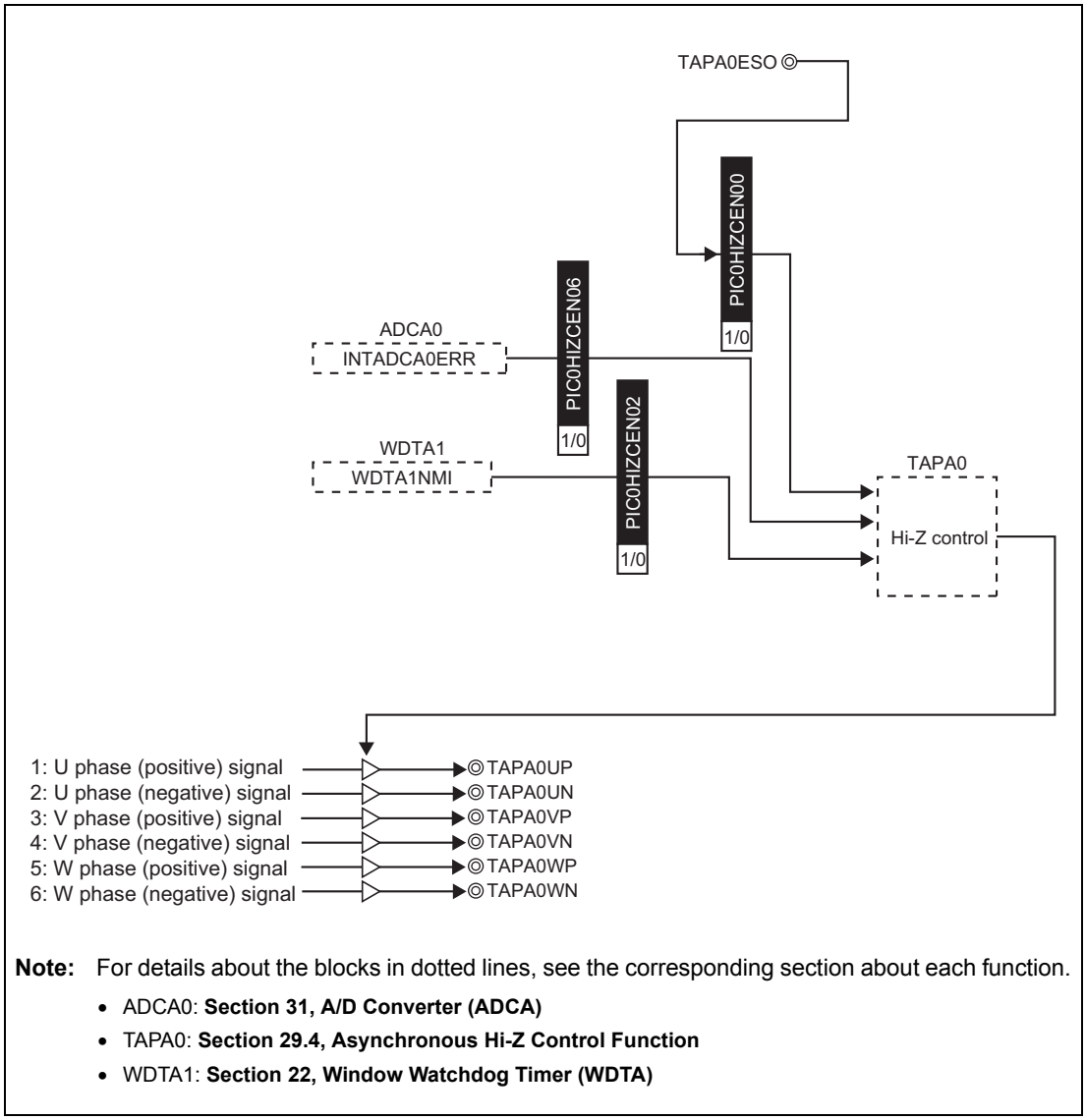
### (2) Block diagram




Note 1. SR flip-flop circuit

Note 2. For the connection destination, see **Figure 29.25, Block Diagram: Motor Output Buffer Control**.

Figure 29.24 Block Diagram: Three-Phase PWM Output with Dead Time




**Note:** For details about the blocks in dotted lines, see the corresponding section about each function.

- ADCA0: **Section 31, A/D Converter (ADCA)**
- TAPA0: **Section 29.4, Asynchronous Hi-Z Control Function**
- WDTA1: **Section 22, Window Watchdog Timer (WDTA)**

**Figure 29.25 Block Diagram: Motor Output Buffer Control**

## 29.11.3 Registers

### 29.11.3.1 PIC0REG2n2 — Timer I/O Control Register 2n2 (n = 0)

**Access:** This register can be read or written in 32-bit units.

**Address:** PIC0REG202: FFDD 00C8$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | PIC0REG2n227 | — | — | — | PIC0REG2n223 | — | — | — | PIC0REG2n219 | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R | R | R | R/W | R | R | R | R/W | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | PIC0REG2n204 | PIC0REG2n203 | PIC0REG2n202 | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R | R |

**Table 29.43 PIC0REG2n2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | Reserved | *1 |
| 27 | PIC0REG2n227 | Selects the signal input to TAUDTTIN15.<br>**PIC0REG2n227** / **Input signal**<br>1 / Signal selected by the PIC0REG2n204 bit.<br>Other than above / Setting prohibited |
| 26 to 24 | Reserved | *1 |
| 23 | PIC0REG2n223 | Selects the signal input to TAUDTTIN13.<br>**PIC0REG2n223** / **Input signal**<br>1 / Signal selected by the PIC0REG2n203 bit.<br>Other than above / Setting prohibited |
| 22 to 20 | Reserved | *1 |
| 19 | PIC0REG2n219 | Selects the signal input to TAUDTTIN11.<br>**PIC0REG2n219** / **Input signal**<br>1 / Signal selected by the PIC0REG2n202 bit<br>Other than above / Setting prohibited |
| 18 to 5 | Reserved | *1 |
| 4 | PIC0REG2n204 | Selects the TIN input signal to TAUDTTIN15.<br>0: Setting prohibited<br>1: Select the set/clear output according to INTTAUDnI8 and INTTAUDnI9. |
| 3 | PIC0REG2n203 | Selects the TIN input signal to TAUDTTIN13.<br>0: Setting prohibited<br>1: Select the set/clear output according to INTTAUDnI6 and INTTAUDnI7. |
| 2 | PIC0REG2n202 | Selects the TIN input signal to TAUDTTIN11.<br>0: Setting prohibited<br>1: Select the set/clear output according to INTTAUDnI4 and INTTAUDnI5. |
| 1, 0 | Reserved | *1 |

Note 1. Some of the bits defined as 0 in the PIC0REG2n2 register are defined for the other timer connection functions. For such bits, use the bit definition of those timer connection functions.

### 29.11.3.2 PIC0HIZCENn — Hi-Z Output Control Register n (n = 0)

The PIC0HIZCENn register selects the Hi-Z output control input signal of TAPAn.

**Access:** This register can be read or written in 8-bit units.

**Address:** <PIC0_base> + $80_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | PIC0HIZCENn6 | — | — | — | PIC0HIZCENn2 | — | PIC0HIZCENn0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R | R | R | R/W | R | R/W |

**Table 29.44 PIC0HIZCENn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 6 | PIC0HIZCENn6 | Selects whether to enable or disable Hi-Z output control by the INTADCA0ERR interrupt signal.<br>0: Disable<br>1: Enable |
| 5 to 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 | PIC0HIZCENn2 | Selects whether to enable or disable Hi-Z output control by the WDTA1NMI interrupt signal.<br>0: Disable<br>1: Enable |
| 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | PIC0HIZCENn0 | Selects whether to enable or disable Hi-Z output control by the TAPAnESO pin input.<br>0: Disable<br>1: Enable |

## 29.11.4 Operation Example

This example shows how to generate each of the set signals and clear signals once or less per cycle and then use the results to output a three-phase PWM waveform with dead time.

This is achieved by combining the following TAUD features:

- PWM output
- One-phase PWM output

In addition, the following peripheral interconnections are used to create the PWM waveform supplied from the set and clear signals generated during PWM output to the input TINm signal (m = 11, 13, or 15) of one-phase PWM output:

- SR flip-flop circuit

Three-phase PWM output is achieved by assigning the one-phase PWM output with dead time achieved using the above features to the U, V, and W phases. Therefore, the set and clear signals of PWM output can be freely specified for each PWM phase. Because the only difference among phases is the assigned channel, only one phase (the U phase) is described below.

### 29.11.4.1 Pwm Output

PWM output uses a combination of CH2, CH4, and CH5.

By specifying the cycle for CDR02, the U phase set value for CDR04, and the U phase clear value for CDR05, a set/clear signal is generated for the SR flip-flop circuit that generates the input TIN11 signal of one-phase PWM output from INT04 and INT05.

Instead of CH4 and CH5, which are used for the above described U phase set/clear signal generation, the V phase uses CH6 and CH7, and the W phase uses CH8 and CH9.

### 29.11.4.2 One-Phase PWM Output

One-phase PWM output is generated from TOUT10 and TOUT11 by using a combination of CH10 and CH11.

By specifying the dead time value for CDR11, a one-phase PWM signal with dead time is output for the TIN11 input.

Similarly, the V phase uses CH12 and CH13 to output a one-phase PWM signal with dead time, while the W phase uses CH14 and CH15.

**CAUTION**

**Specify the same clock for each TAUDn channel that uses the PWM output and one-phase PWM output features.**

For details about the TAUD functions, see **Section 25, Timer Array Unit D (TAUD)**.

### 29.11.4.3 SR Flip-Flop Circuit

The PWM waveform supplied to input TIN11 of one-phase PWM output is generated by using the U phase set signal generated by CH4 of TAUD and the U phase clear signal generated by CH5.

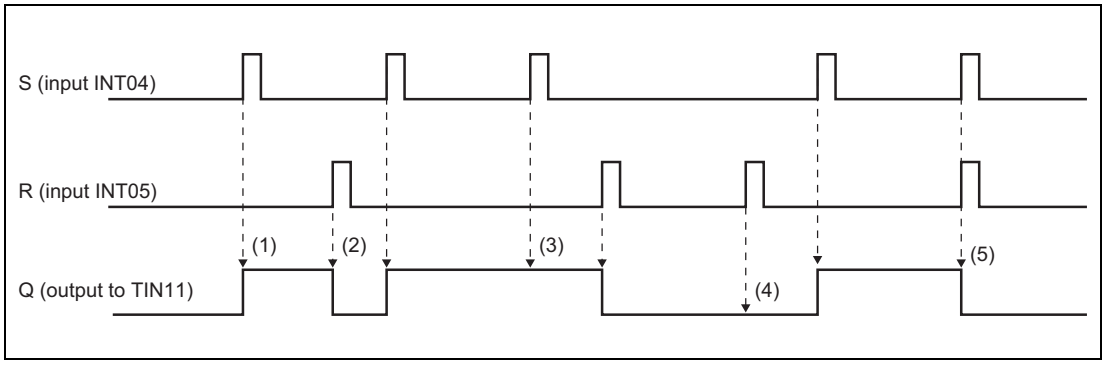



**Figure 29.26 SR Flip-Flop Circuit Operation Timing Chart (U phase example)**

(1) When a signal is input to input S, output Q goes to the high level at the rising edge of S.

(2) When a signal is input to input R, output Q goes to the low level at the rising edge of R.

(3) If a signal is input to input S while output Q is at the high level, output Q is not affected.

(4) If a signal is input to input R while output Q is at the low level, output Q is not affected.

(5) If a signal is input to input S and input R at the same time, input R is prioritized and output Q goes to the low level at the rising edge of R.

The V phase uses INT06 and INT07 as input to supply a PWM waveform to TIN13, and the W phase uses INT08 and INT09 as input to supply a PWM waveform to TIN15.

The output change timing of the PWM waveform generated during one-phase PWM output is based on PWM output.

The active level output timing set signal and inactive level output timing clear signal of PWM are generated during PWM output. By inputting these signals to the SR flip-flop circuit, a PWM signal that can be changed at any time is generated.

A one-phase PWM signal is output by generating a positive or negative PWM waveform and then adding dead time to it according to changes in the generated PWM signal.

PIC is used to set/clear signal generated during PWM output as the TIN input for one-phase PWM output through the SR flip-flop circuit.

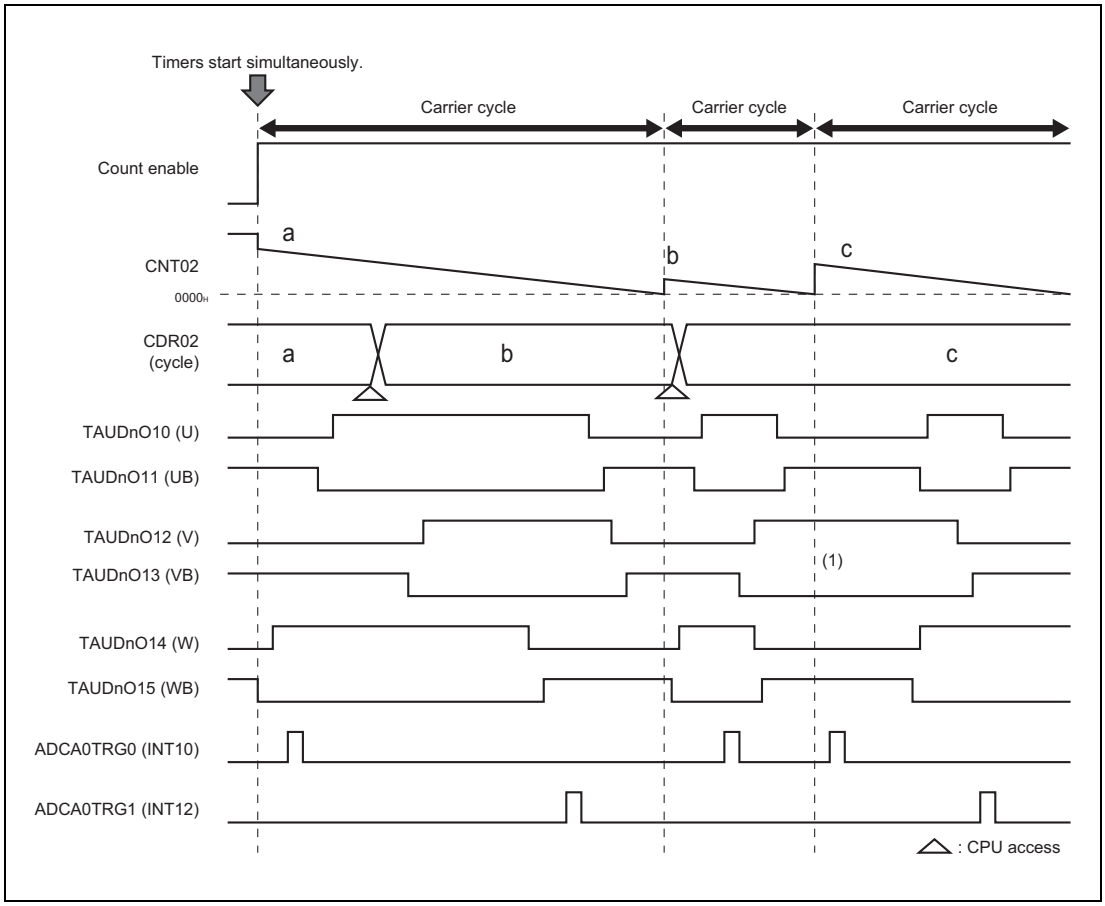



**Figure 29.27 Example of Three-Phase PWM (U/UB, V/VB, W/WB) Output with Dead Time**

**Figure 29.27** shows a typical example of three-phase PWM output with dead time.

By appropriately setting up the set/clear signal output timing, PWM output that extends across carrier cycles (point (1)) and other types of output are also possible.

In this example, ADCA0TRG0 and ADCA0TRG1 (which are at the bottom) use the CNT and INT signals of CH10 and CH12, which are not used for one-phase PWM output, and the A/D trigger signal is output by performing type-1 A/D trigger output.

In this way, because only the TOUTm signal that performs signal output for the channel performing positive phase output is used during one-phase PWM output, any feature that uses CNTm, CDRm, or INTm can be specified. For details, see **Section 25, Timer Array Unit D (TAUD)** (m = 10, 12, or 14).

The following figures show timing charts for outputting a three-phase PWM signal with dead time.

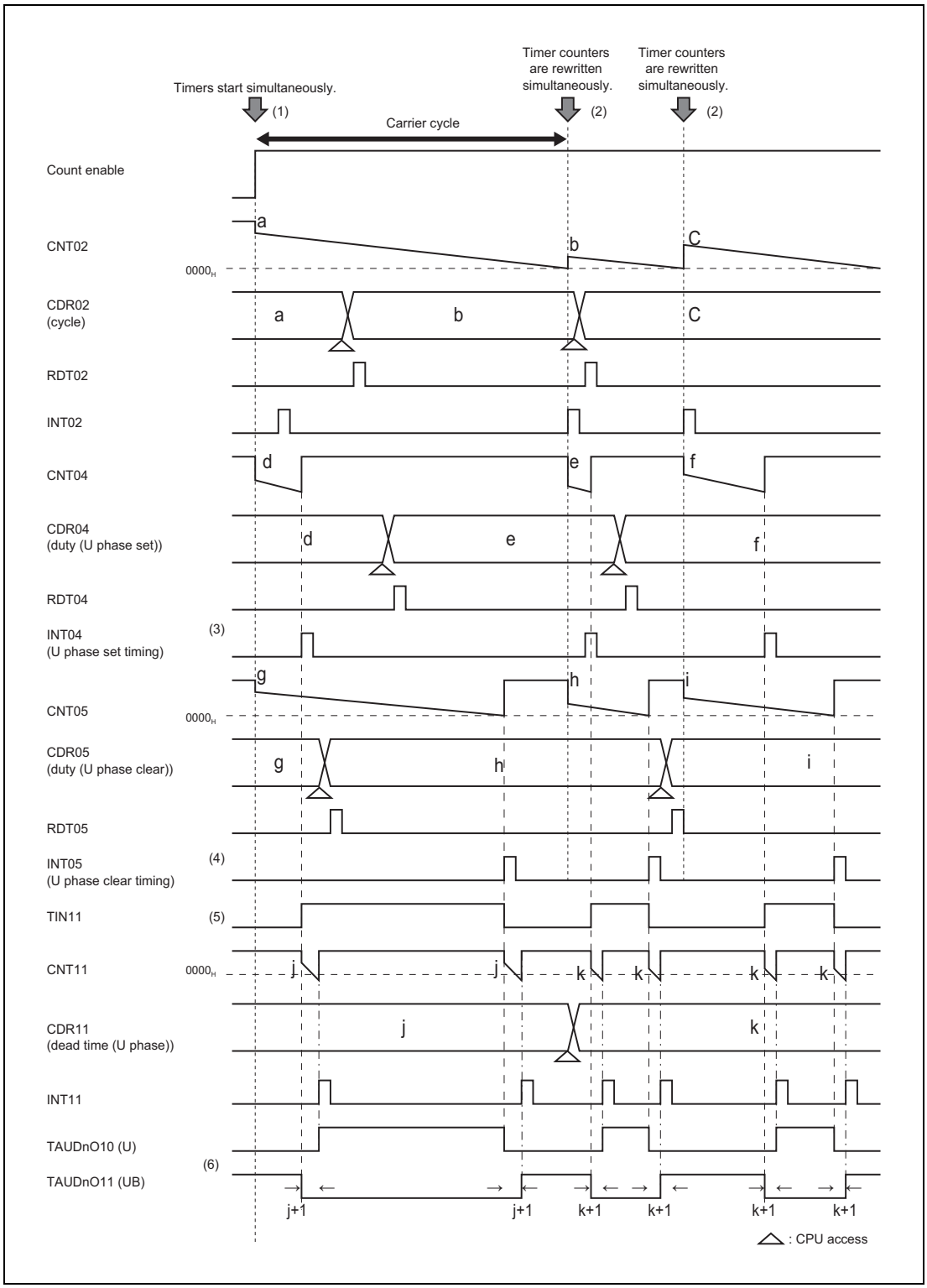



**Figure 29.28 Example of One-Phase PWM (U phase, UB phase) Output with Dead Time**

An operation example of the timer configuration for performing the U phase PWM output in **Figure 29.28** is provided below.

(1) By simultaneously starting timers, CH2 (the carrier cycle timer), CH4 (the U phase set signal output timing timer), and CH5 (the U phase clear signal output timing timer) are started simultaneously.
The CH11 timer is also enabled, but until a TIN11 edge is detected, which is the count start timing, counting is not performed.

(2) For CH4 and CH5, when there is a CH2 underflow, the settings from CDR04 and CDR05 are reloaded to CNT04 and CNT05.

(3) When there is a CH4 underflow, the U phase set timing signal (INT04) is generated.

(4) When there is a CH5 underflow, the U phase clear timing signal (INT05) is generated.

(5) The peripheral interconnections supply the output of the SR flip-flop circuit that uses INT04 (the set timing signal) and INT05 (the clear timing signal) as input to the input TIN11 signal of one-phase PWM output.

(6) During one-phase PWM output, a PWM waveform with dead time is output by detecting a TIN11 edge.

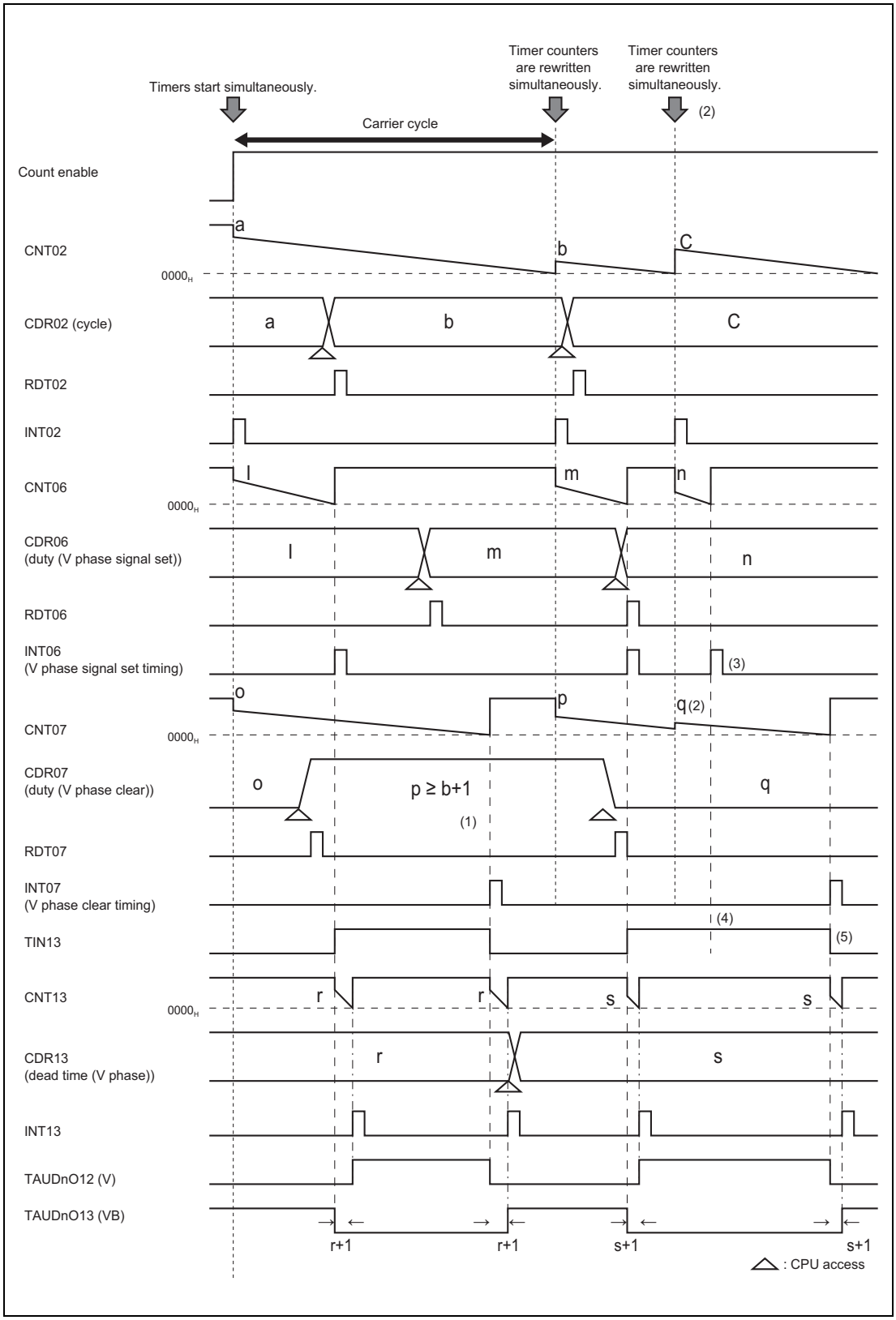



**Figure 29.29 Example of One-Phase PWM (V phase, VB phase) Output with Dead Time**

An operation example of the timer configuration for performing the V phase PWM output in **Figure 29.29** is provided below.

For details about the operations from when timers are simultaneously started until a one-phase PWM signal is output, see the U phase operation example.

(1) If the setting of CH7 (the V phase clear signal output timing timer), which generates the V phase clear timing signal (INT07), is greater than the CH2 (the carrier cycle timer) setting.

(2) Before a V phase clear timing signal (INT07) is generated by a CH7 underflow, a CH2 (carrier cycle timer) underflow occurs, and the CH7 setting is reloaded.

(3) It causes the V phase clear timing signal (INT07) not to be generated, resulting in consecutive generation of the V phase set timing signal (INT06).

(4) In this case, because the V phase set timing signal (INT06) is ignored by the SR flip-flop circuit, there is no effect on the PWM output waveform. Therefore, a PWM waveform that extends across carrier cycles is output.

(5) The PWM output is changed at the timing of the next V phase clear timing signal (INT07).

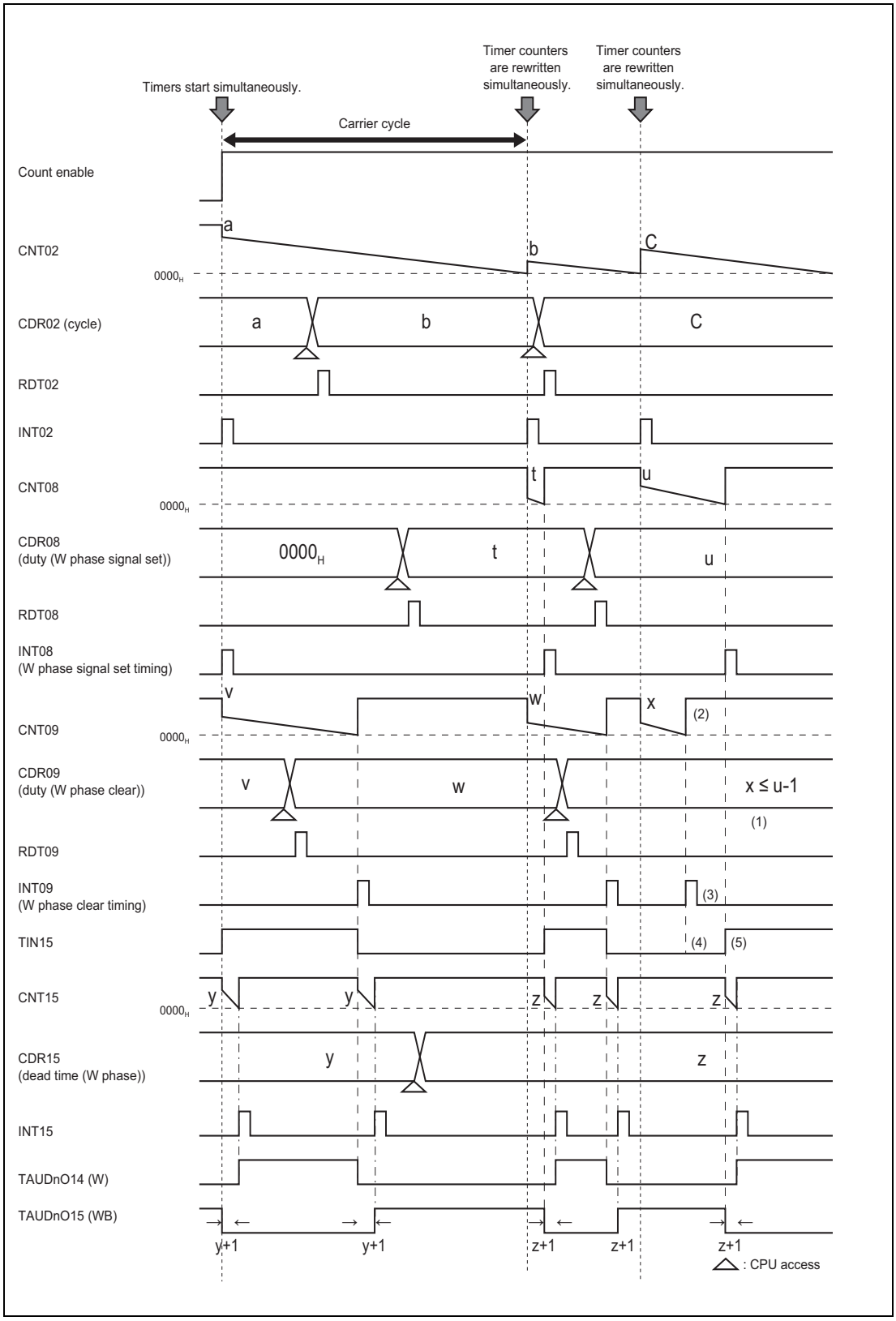



**Figure 29.30 Example of One-Phase PWM (W phase, WB phase) Output with Dead Time**

An operation example of the timer configuration for performing the W phase PWM output in **Figure 29.30** is provided below.

For details about the operations from when timers are simultaneously started until a one-phase PWM signal is output, see the U phase operation example.

(1) If the setting of CH9 (the W phase clear signal output timing timer), which generates the W phase clear timing signal (INT09), is less than the CH8 (the W phase set signal output timing timer) setting.

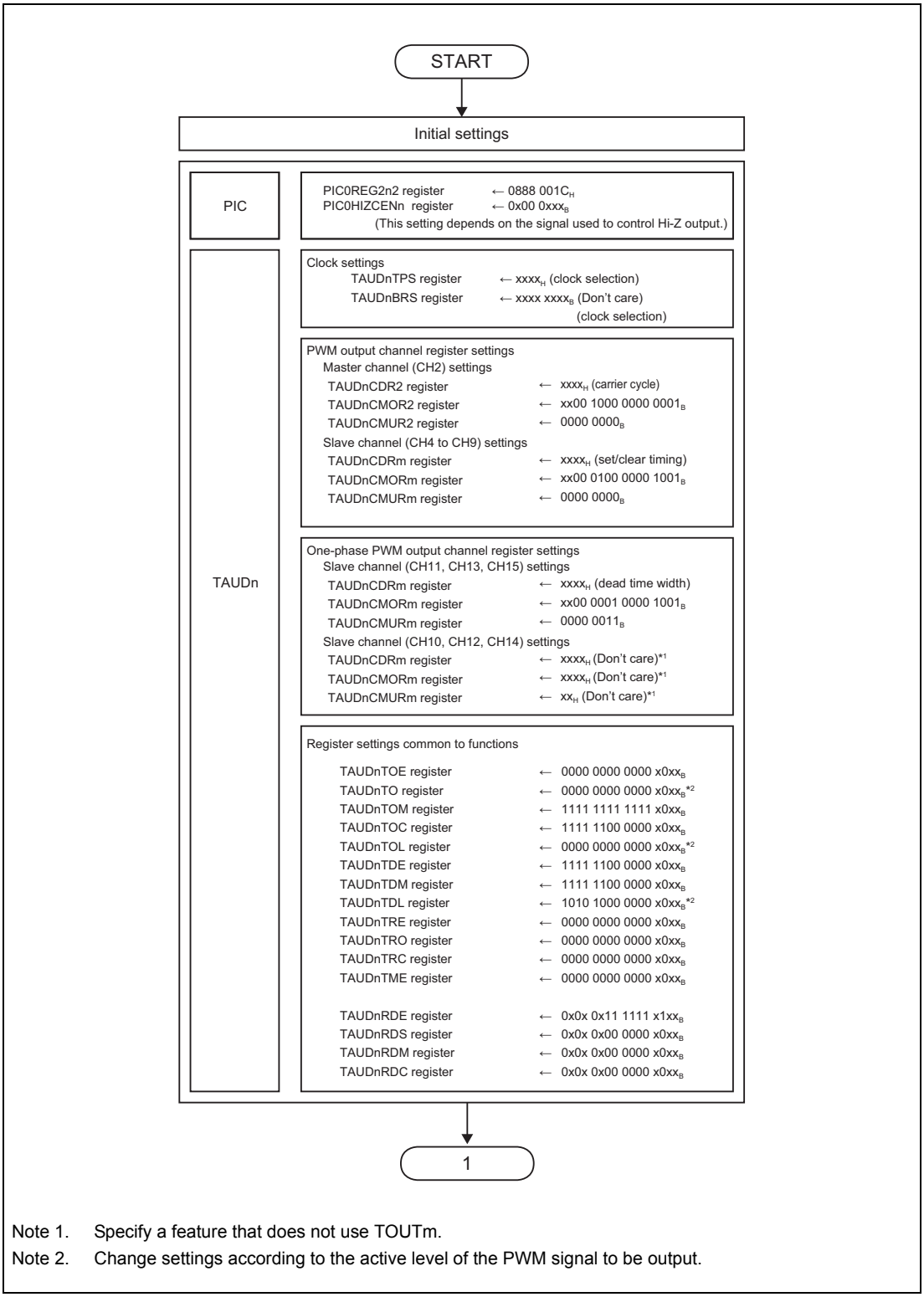
(2) Before a W phase set timing signal (INT08) is generated by a CH8 underflow, a CH9 (W phase clear signal output timing timer) underflow occurs, and the W phase clear timing signal (INT09) is generated.

(3) This results in consecutive W phase clear timing signals (INT09) being generated.

(4) In this case, because the consecutively generated W phase clear timing signals (INT09) are ignored by the SR flip-flop circuit, there is no effect on the PWM output waveform.

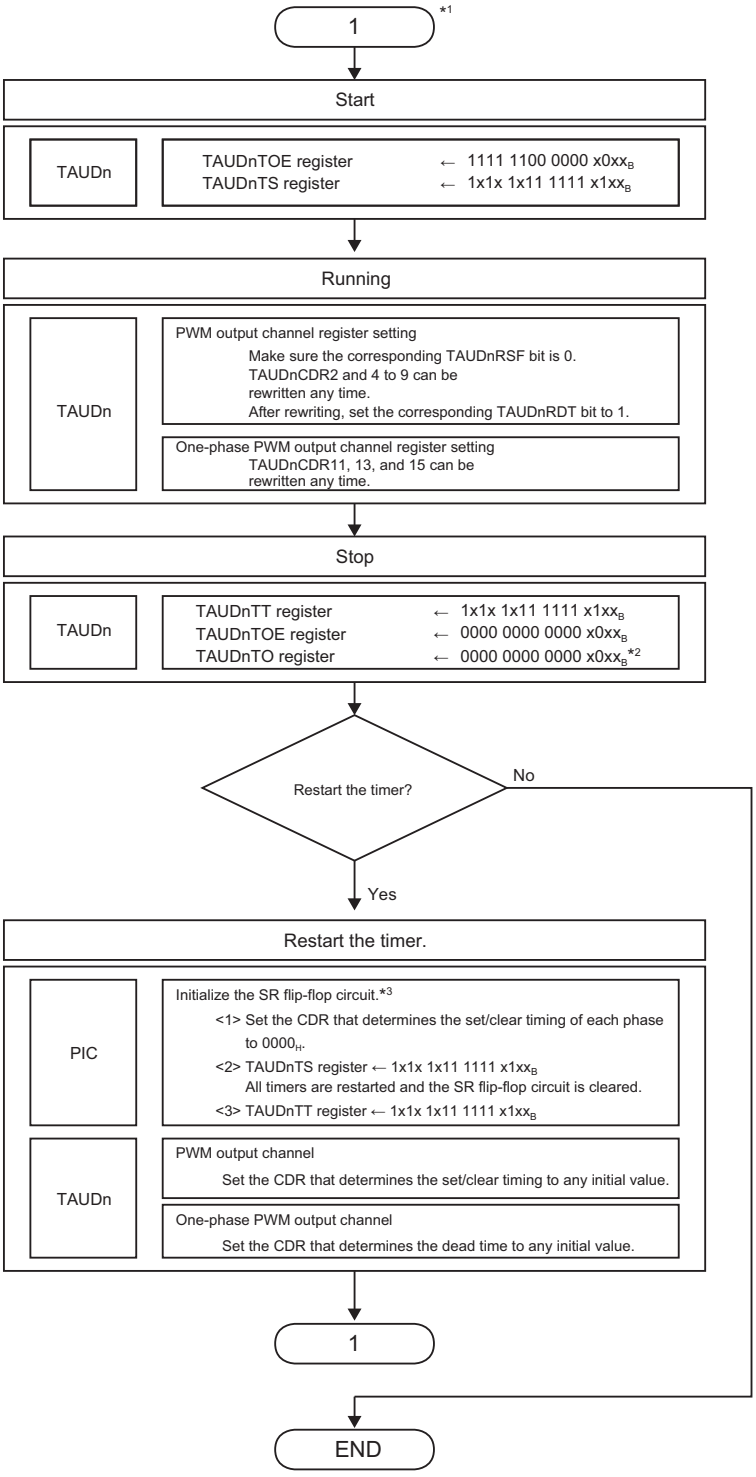
(5) The PWM output is changed at the timing of the next W phase set timing signal (INT08).

### 29.11.5 Setup Flow




Note 1. Specify a feature that does not use TOUTm.
Note 2. Change settings according to the active level of the PWM signal to be output.

**Figure 29.31 Setup Flow (Active High Example)**




Note 1. Specify the selection register and output port to use after specifying the initial settings for the peripheral interconnections and timers.

Note 2. Change settings according to the active level of the PWM signal to be output.

Note 3. If initialization processing is not performed, the SR flip-flop circuit enters the timer-stopped state, and the timer restart output pulse might be output at an unintended level.

**Figure 29.32 Setup Flow (Active High Example) (continued)**

## 29.11.6 Setting Examples for Operation Functions

This section provides example settings for each register.

### 29.11.6.1 TAUDn Settings (Active High Example)

**Table 29.45 TAUDn: CH2-related (PWM Output Master Channel*1)**

| Register | Bit Position | Bit Name | Setting | Remark |
|---|---|---|---|---|
| TAUDnCMOR2 | 15, 14 | TAUDnCKS[1:0] | Don't care*2 | Operation clock setting |
| | 13, 12 | TAUDnCCS[1:0] | 00 | |
| | 11 | TAUDnMAS | 1 | |
| | 10 to 8 | TAUDnSTS[2:0] | 000 | |
| | 7, 6 | TAUDnCOS[1:0] | 00 | |
| | 5 | | 0 | Fixed to 0 |
| | 4 to 1 | TAUDnMD[4:1] | 0000 | |
| | 0 | TAUDnMD0 | 1 | |
| TAUDnCMUR2 | 1, 0 | TAUDnTIS[1:0] | 00 | |

Note 1. The master channel and slave channel names are defined for TAUD PWM output. For details, see **Section 25, Timer Array Unit D (TAUD)**.

Note 2. The same operation clock must be specified for the master channel and slave channel.

**Table 29.46 TAUDn: CH4 to CH9-related (PWM Output Slave Channel*1) (m = 4 to 9)**

| Register | Bit Position | Bit Name | Setting | Remark |
|---|---|---|---|---|
| TAUDnCMORm | 15, 14 | TAUDnCKS[1:0] | Any*2 | Operation clock setting |
| | 13, 12 | TAUDnCCS[1:0] | 00 | |
| | 11 | TAUDnMAS | 0 | |
| | 10 to 8 | TAUDnSTS[2:0] | 100 | |
| | 7, 6 | TAUDnCOS[1:0] | 00 | |
| | 5 | | 0 | Fixed to 0 |
| | 4 to 1 | TAUDnMD[4:1] | 0100 | |
| | 0 | TAUDnMD0 | 1 | |
| TAUDnCMURm | 1, 0 | TAUDnTIS[1:0] | 00 | |

Note 1. The master channel and slave channel names are defined for TAUD PWM output. For details, see **Section 25, Timer Array Unit D (TAUD)**.

Note 2. The same operation clock must be specified for the master channel and slave channel.

**NOTE**

For the TAUDnCMORm register used during PWM output, TAUDnCKS[1:0] (which selects the operation clock) can be set to any value, but other control bits have fixed values. For details, see **Section 25, Timer Array Unit D (TAUD)**.

**Table 29.47 TAUDn: CH11, CH13, and CH15-related (One-phase PWM Output) (m = 11, 13, or 15)**

| Register | Bit Position | Bit Name | Setting | Remark |
|---|---|---|---|---|
| TAUDnCMORm | 15, 14 | TAUDnCKS[1:0] | Don't care*1 | Operation clock setting |
| | 13, 12 | TAUDnCCS[1:0] | 00 | |
| | 11 | TAUDnMAS | 0 | |
| | 10 to 8 | TAUDnSTS[2:0] | 001 | |
| | 7, 6 | TAUDnCOS[1:0] | 00 | |
| | 5 | | 0 | Fixed to 0 |
| | 4 to 1 | TAUDnMD[4:1] | 0100 | |
| | 0 | TAUDnMD0 | 1 | |
| TAUDnCMURm | 1, 0 | TAUDnTIS[1:0] | 11 | Both rising and falling TINm edges are detected as valid. (High width) |

Note 1. Specify the same operation clock settings as for the PWM output master channel (CH2).

**NOTE**

For the TAUDnCMORm register used during one-phase PWM output, TAUDnCKS[1:0] (which selects the operation clock) can be set to any value, but other control bits have fixed values. CH10, CH12, and CH14 can be used with any feature that does not use TOUTm output (such as A/D trigger output). For details, see **Section 25, Timer Array Unit D (TAUD)**.

**Table 29.48 Common TAUDn Channel Settings (1/4)**

| Register | Bit Position | Bit Name | Setting | Remark |
|---|---|---|---|---|
| TAUDnTOE | 15 to 10 | TAUDnTOE15 to TAUDnTOE10 | 0<br>1 | Disable the timer.<br>Enable the timer. |
| | 9 to 4 | TAUDnTOE09 to TAUDnTOE04 | 0 | These are fixed to 0 because TOUT09 to TOUT04 are not used. |
| | 3 | TAUDnTOE03 | Don't care | |
| | 2 | TAUDnTOE02 | 0 | This is fixed to 0 because TOUT02 is not used. |
| | 1, 0 | TAUDnTOE01 TAUDnTOE00 | Don't care | |
| TAUDnTO | 15 to 10 | TAUDnTO15 to TAUDnTO10 | 0*1 | Output a low-level signal to TOUT15 to TOUT10. |
| | 9 to 4 | TAUDnTO09 to TAUDnTO04 | 0 | Output a low-level signal to TOUT09 to TOUT04. |
| | 3 | TAUDnTO03 | Don't care | |
| | 2 | TAUDnTO02 | 0 | Output a low-level signal to TOUT02. |
| | 1, 0 | TAUDnTO01 TAUDnTO00 | Don't care | |
| TAUDnTOM | 15 to 4 | TAUDnTOM15 to TAUDnTOM04 | 1 | Synchronous operation mode |
| | 3 | TAUDnTOM03 | Don't care | |
| | 2 | TAUDnTOM02 | 0 | Independent operation mode |
| | 1, 0 | TAUDnTOM01 TAUDnTOM00 | Don't care | |

Table 29.48 Common TAUDn Channel Settings (2/4)

| Register | Bit Position | Bit Name | Setting | Remark |
|---|---|---|---|---|
| TAUDnTOC | 15 to 10 | TAUDnTOC15 to TAUDnTOC10 | 1 | Synchronous operation mode 2 |
| | 9 to 4 | TAUDnTOC09 to TAUDnTOC04 | 0 | Synchronous operation mode 1 |
| | 3 | TAUDnTOC03 | Don't care | |
| | 2 | TAUDnTOC02 | 0 | Operation mode 1 |
| | 1, 0 | TAUDnTOC01 TAUDnTOC00 | Don't care | |
| TAUDnTOL | 15 to 4 | TAUDnTOL15 to TAUDnTOL04 | 0*1 | Positive logic output (active high) |
| | 3 | TAUDnTOL03 | Don't care | |
| | 2 | TAUDnTOL02 | 0 | Positive logic output (active high) |
| | 1, 0 | TAUDnTOL01 TAUDnTOL00 | Don't care | |
| TAUDnTDE | 15 to 10 | TAUDnTDE15 to TAUDnTDE10 | 1 | Enable dead time control.*2 |
| | 9 to 4 | TAUDnTDE09 to TAUDnTDE04 | 0 | Disable dead time control. |
| | 3 | TAUDnTDE03 | Don't care | |
| | 2 | TAUDnTDE02 | 0 | Disable dead time control. |
| | 1, 0 | TAUDnTDE01 TAUDnTDE00 | Don't care | |
| TAUDnTDM | 15 to 10 | TAUDnTDM15 to TAUDnTDM10 | 1 | Output dead time upon detecting a TINm input edge at a lower odd channel. |
| | 9 to 4 | TAUDnTDM09 to TAUDnTDM04 | 0 | Invalid because dead time control is disabled. |
| | 3 | TAUDnTDM03 | Don't care | |
| | 2 | TAUDnTDM02 | 0 | Invalid because dead time control is disabled. |
| | 1, 0 | TAUDnTDM01 TAUDnTDM00 | Don't care | |
| TAUDnTDL | 15 | TAUDnTDL15 | 1*1 | Dead time is in the negative segment of the W phase output |
| | 14 | TAUDnTDL14 | 0*1 | Dead time is in the positive segment of the W phase output |
| | 13 | TAUDnTDL13 | 1*1 | Dead time is in the negative segment of the V phase output |
| | 12 | TAUDnTDL12 | 0*1 | Dead time is in the positive segment of the V phase output |
| | 11 | TAUDnTDL11 | 1*1 | Dead time is in the negative segment of the U phase output |
| | 10 | TAUDnTDL10 | 0*1 | Dead time is in the positive segment of the U phase output |
| | 9 to 4 | TAUDnTDL09 to TAUDnTDL04 | 0 | Invalid because dead time control is disabled. |
| | 3 | TAUDnTDL03 | Don't care | |
| | 2 | TAUDnTDL02 | 0 | Invalid because dead time control is disabled. |
| | 1, 0 | TAUDnTDL01 TAUDnTDL00 | Don't care | |

Table 29.48 Common TAUDn Channel Settings (3/4)

| Register | Bit Position | Bit Name | Setting | Remark |
|---|---|---|---|---|
| TAUDnTRE | 15 to 4 | TAUDnTRE15 to TAUDnTRE04 | 0 | Stop real-time output. |
| | 3 | TAUDnTRE03 | Don't care | |
| | 2 | TAUDnTRE02 | 0 | Stop real-time output. |
| | 1, 0 | TAUDnTRE01 TAUDnTRE00 | Don't care | |
| TAUDnTRO | 15 to 4 | TAUDnTRO15 to TAUDnTRO04 | 0 | Invalid because real-time output is disabled. |
| | 3 | TAUDnTRO03 | Don’t care | |
| | 2 | TAUDnTRO02 | 0 | Invalid because real-time output is disabled. |
| | 1, 0 | TAUDnTRO01 TAUDnTRO00 | Don’t care | |
| TAUDnTRC | 15 to 4 | TAUDnTRC15 to TAUDnTRC04 | 0 | Do not use this channel to generate the real-time output trigger. |
| | 3 | TAUDnTRC03 | Don’t care | |
| | 2 | TAUDnTRC02 | 0 | Do not use this channel to generate the real-time output trigger. |
| | 1, 0 | TAUDnTRC01 TAUDnTRC00 | Don’t care | |
| TAUDnTME | 15 to 4 | TAUDnTME15 to TAUDnTME04 | 0 | Disable modulation output for timer output and real-time output. |
| | 3 | TAUDnTME03 | Don’t care | |
| | 2 | TAUDnTME02 | 0 | Disable modulation output for timer output and real-time output. |
| | 1, 0 | TAUDnTME01 TAUDnTME00 | Don’t care | |
| TAUDnRDE | 15 | TAUDnRDE15 | 0 | Disable simultaneous rewriting. |
| | 14 | TAUDnRDE14 | Don’t care | |
| | 13 | TAUDnRDE13 | 0 | Disable simultaneous rewriting. |
| | 12 | TAUDnRDE12 | Don’t care | |
| | 11 | TAUDnRDE11 | 0 | Disable simultaneous rewriting. |
| | 10 | TAUDnRDE10 | Don’t care | |
| | 9 to 4 | TAUDnRDE09 to TAUDnRDE04 | 1 | Enable simultaneous rewriting. |
| | 3 | TAUDnRDE03 | Don’t care | |
| | 2 | TAUDnRDE02 | 1 | Enable simultaneous rewriting. |
| | 1, 0 | TAUDnRDE01 TAUDnRDE00 | Don’t care | |

Table 29.48 Common TAUDn Channel Settings (4/4)

| Register | Bit Position | Bit Name | Setting | Remark |
|---|---|---|---|---|
| TAUDnRDS | 15 | TAUDnRDS15 | 0 | Do not enable simultaneous rewriting by using another upper channel. |
| | 14 | TAUDnRDS14 | Don't care | |
| | 13 | TAUDnRDS13 | 0 | Do not enable simultaneous rewriting by using another upper channel. |
| | 12 | TAUDnRDS12 | Don't care | |
| | 11 | TAUDnRDS11 | 0 | Do not enable simultaneous rewriting by using another upper channel. |
| | 10 | TAUDnRDS10 | Don't care | |
| | 9 to 4 | TAUDnRDS09 to TAUDnRDS04 | 0 | Enable simultaneous rewriting by using a master channel. |
| | 3 | TAUDnRDS03 | Don't care | |
| | 2 | TAUDnRDS02 | 0 | Enable simultaneous rewriting by using a master channel. |
| | 1, 0 | TAUDnRDS01 TAUDnRDS00 | Don't care | |
| TAUDnRDM | 15 | TAUDnRDM15 | 0 | Invalid because simultaneous rewriting is not enabled. |
| | 14 | TAUDnRDM14 | Don't care | |
| | 13 | TAUDnRDM13 | 0 | Invalid because simultaneous rewriting is not enabled. |
| | 12 | TAUDnRDM12 | Don't care | |
| | 11 | TAUDnRDM11 | 0 | Invalid because simultaneous rewriting is not enabled. |
| | 10 | TAUDnRDM10 | Don't care | |
| | 9 to 4 | TAUDnRDM09 to TAUDnRDM04 | 0 | Load the signal when the master channel starts counting. |
| | 3 | TAUDnRDM03 | Don't care | |
| | 2 | TAUDnRDM02 | 0 | Load the signal when the master channel starts counting. |
| | 1, 0 | TAUDnRDM01 TAUDnRDM00 | Don't care | |
| TAUDnRDC | 15 | TAUDnRDC15 | 0 | Invalid because simultaneous rewriting is not enabled. |
| | 14 | TAUDnRDC14 | Don't care | |
| | 13 | TAUDnRDC13 | 0 | Invalid because simultaneous rewriting is not enabled. |
| | 12 | TAUDnRDC12 | Don't care | |
| | 11 | TAUDnRDC11 | 0 | Invalid because simultaneous rewriting is not enabled. |
| | 10 | TAUDnRDC10 | Don't care | |
| | 9 to 4 | TAUDnRDC09 to TAUDnRDC04 | 0 | Do not use this channel to generate the simultaneous rewrite trigger. |
| | 3 | TAUDnRDC03 | Don't care | |
| | 2 | TAUDnRDC02 | 1 | Do not use this channel to generate the simultaneous rewrite trigger. |
| | 1, 0 | TAUDnRDC01 TAUDnRDC00 | Don't care | |

Note 1. Change the setting according to the used system.

Note 2. These are used to control positive/negative phase waveform output for which even channels are paired with odd channels to perform dead time control. For details, see **Section 25, Timer Array Unit D (TAUD)**.

### 29.11.6.2 PIC Settings

**Table 29.49 PIC Settings**

| Register | Bit Position | Bit Name | Setting | Remark |
|---|---|---|---|---|
| PIC0REG2n2 | 27 | PIC0REG2n227 | 1 | Select the input selected by the PIC0REG2n204 bit. |
| | 23 | PIC0REG2n223 | 1 | Select the input selected by the PIC0REG2n203 bit. |
| | 19 | PIC0REG2n219 | 1 | Select the input selected by the PIC0REG2n202 bit. |
| | 4 | PIC0REG2n204 | 1 | Select the set/clear output according to INTTAUDnI8 and INTTAUDnI9. |
| | 3 | PIC0REG2n203 | 1 | Select the set/clear output according to INTTAUDnI6 and INTTAUDnI7. |
| | 2 | PIC0REG2n202 | 1 | Select the set/clear output according to INTTAUDnI4 and INTTAUDnI5. |

# 29.12 High-accuracy Triangle PWM Output with Dead Time

## 29.12.1 Functional Overview

Compared to the triangle PWM output with dead time of TAUD, this makes it possible to control the variable dead time areas near duties of 100% and 0%. This makes more accurate triangle PWM output possible.

For the triangle PWM output with dead time feature of TAUD, it is not possible to output a UB phase dead time pulse, such as when transitioning to U phase 0% triangular wave output. (See **Figure 29.33**)

For this feature, a pulse is generated in combination with the TAUD timer output, and a pseudo dead time pulse is added.

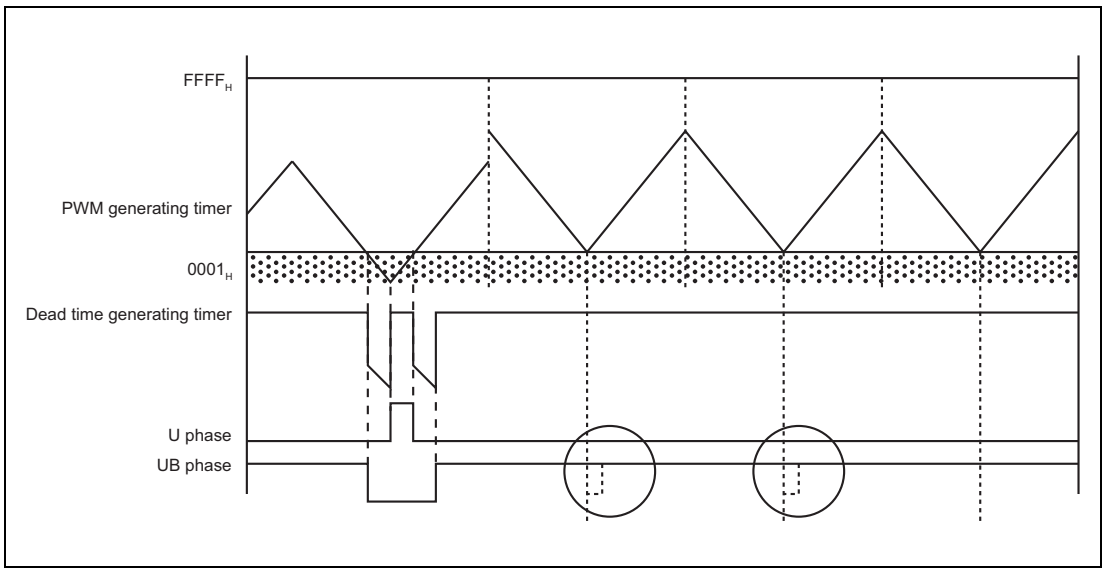



**Figure 29.33 Timing of Dead Time Output by the TAUD Feature for Outputting a Triangle PWM Signal with Dead Time**

## 29.12.2 Configuration

The unit and channel configuration for this feature is shown below. (n = 0)

**Table 29.50 Configuration of Delay Pulse Output with Dead Time**

| Timer | Timer Motor Control Function |
|---|---|
| TAUD0 CH2, CH4 to CH15 (used channels fixed) | TAPA0 |

The signal names used in the descriptions below are abbreviations. The actual signal names corresponding to each abbreviation are as follows:

- INTm → INTTAUDnIm (TAUDn channel m interrupt)
- TINm → TAUDTTINm (TAUDn channel m input)
- TOUTm → TAUDTTOUTm (TAUDn channel m output)
- CDRm → TAUDnCDRm (TAUDn channel m data register)
- CNTm → TAUDnCNTm (TAUDn channel m counter register)

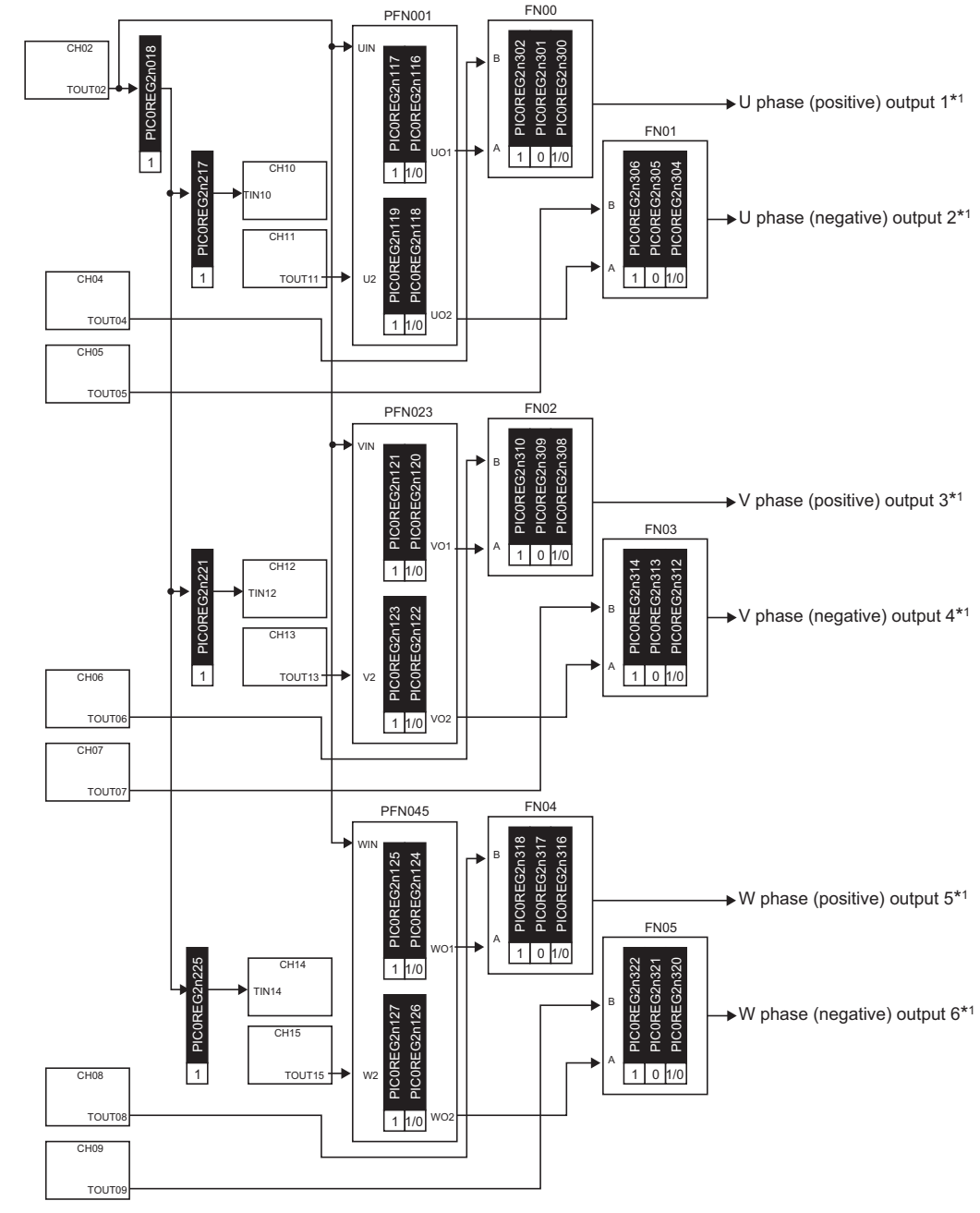
### (1) TAUDn configuration

**Table 29.51 TAUD Configuration**

| CH | Function Name | M/S*1 | CDR Setting | Description |
|---|---|---|---|---|
| 2 | Triangle PWM output with dead time (CH2 is the master channel for CH4 to CH9.) | M | Cycle | |
| 4 | | S | Duty (U phase) | |
| 5 | | S | Dead time (U phase) | |
| 6 | | S | Duty (V phase) | |
| 7 | | S | Dead time (V phase) | |
| 8 | | S | Duty (W phase) | |
| 9 | | S | Dead time (W phase) | |
| 10 | One-shot pulse output | M | Delay | Generate the pulse to be inserted into the variable dead time area for U phase PWM. |
| 11 | | S | Pulse width | |
| 12 | One-shot pulse output | M | Delay | Generate the pulse to be inserted into the variable dead time area for V phase PWM. |
| 13 | | S | Pulse width | |
| 14 | One-shot pulse output | M | Delay | Generate the pulse to be inserted into the variable dead time area for W phase PWM. |
| 15 | | S | Pulse width | |

Note 1. M: Master channel, S: Slave channel

### (2) Block diagram




Note 1. For the connection destination, see **Figure 29.25, Block Diagram: Motor Output Buffer Control**.

**Figure 29.34 Block Diagram: High-Accuracy Triangle PWM Output with Dead Time**

## 29.12.3 Registers

### 29.12.3.1 PIC0REG2n0 — Timer I/O Control Register 2n0 (n = 0)

This register selects TAUDn input.

**Access:** This register can be read or written in 32-bit units.

**Address:** PIC0REG200: FFDD 00C0$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | PIC0REG2n018 | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 29.52 PIC0REG2n0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 19 | Reserved | *1 |
| 18 | PIC0REG2n018 | Select the TIN input signal to TAUDTTIN10, TAUDTTIN12, and TAUDTTIN14.<br>0: Setting prohibited<br>1: Select TAUDTTOUT2. |
| 17 to 0 | Reserved | *1 |

Note 1. Some of the bits defined as 0 in the PIC0REG2n0 register are defined for the other timer connection functions. For such bits, use the bit definition of those timer connection functions.

### 29.12.3.2 PIC0REG2n1 — Timer I/O Control Register 2n1 (n = 0)

This register selects the logic of a combination circuit.

**Access:** This register can be read or written in 32-bit units.

**Address:** PIC0REG201: FFDD 00C4$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | PIC0REG2n127 | PIC0REG2n126 | PIC0REG2n125 | PIC0REG2n124 | PIC0REG2n123 | PIC0REG2n122 | PIC0REG2n121 | PIC0REG2n120 | PIC0REG2n119 | PIC0REG2n118 | PIC0REG2n117 | PIC0REG2n116 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 29.53 PIC0REG2n1 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | Reserved | *1 |
| 27, 26 | PIC0REG2n127<br>PIC0REG2n126 | Select the FN05 A input signal according to the output logic specified for CH9 of TAUDn. |

| PIC0REG2n127 | PIC0REG2n126 | Input Signal |
|---|---|---|
| 1 | 0 | Combination circuit output<br>(Select this when the active high setting is specified (TAUDnTOL09 = 0).) |
| 1 | 1 | Inverted combination circuit output<br>(Select this when the active low setting is specified (TAUDnTOL09 = 1).) |
| Other than above | | Setting prohibited |

| Bit Position | Bit Name | Function |
|---|---|---|
| 25, 24 | PIC0REG2n125<br>PIC0REG2n124 | Select the FN04 A input signal according to the output logic specified for CH8 of TAUDn. |

| PIC0REG2n125 | PIC0REG2n124 | Input Signal |
|---|---|---|
| 1 | 0 | Combination circuit output<br>(Select this when the active high setting is specified (TAUDnTOL08 = 0).) |
| 1 | 1 | Inverted combination circuit output<br>(Select this when the active low setting is specified (TAUDnTOL08 = 1).) |
| Other than above | | Setting prohibited |

| Bit Position | Bit Name | Function |
|---|---|---|
| 23, 22 | PIC0REG2n123<br>PIC0REG2n122 | Select the FN03 A input signal according to the output logic specified for CH7 of TAUDn. |

| PIC0REG2n123 | PIC0REG2n122 | Input Signal |
|---|---|---|
| 1 | 0 | Combination circuit output<br>(Select this when the active high setting is specified (TAUDnTOL07 = 0).) |
| 1 | 1 | Inverted combination circuit output<br>(Select this when the active low setting is specified (TAUDnTOL07 = 1).) |
| Other than above | | Setting prohibited |

**Table 29.53 PIC0REG2n1 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 21, 20 | PIC0REG2n121<br>PIC0REG2n120 | Select the FN02 A input signal according to the output logic specified for CH6 of TAUDn. |

| PIC0REG2n121 | PIC0REG2n120 | Input Signal |
|---|---|---|
| 1 | 0 | Combination circuit output<br>(Select this when the active high setting is specified (TAUDnTOL06 = 0).) |
| 1 | 1 | Inverted combination circuit output<br>(Select this when the active low setting is specified (TAUDnTOL06 = 1).) |
| Other than above | | Setting prohibited |

| Bit Position | Bit Name | Function |
|---|---|---|
| 19, 18 | PIC0REG2n119<br>PIC0REG2n118 | Select the FN01 A input signal according to the output logic specified for CH5 of TAUDn. |

| PIC0REG2n119 | PIC0REG2n118 | Input Signal |
|---|---|---|
| 1 | 0 | Combination circuit output<br>(Select this when the active high setting is specified (TAUDnTOL05 = 0).) |
| 1 | 1 | Inverted combination circuit output<br>(Select this when the active low setting is specified (TAUDnTOL05 = 1).) |
| Other than above | | Setting prohibited |

| Bit Position | Bit Name | Function |
|---|---|---|
| 17, 16 | PIC0REG2n117<br>PIC0REG2n116 | Select the FN00 A input signal according to the output logic specified for CH4 of TAUDn. |

| PIC0REG2n117 | PIC0REG2n116 | Input Signal |
|---|---|---|
| 1 | 0 | Combination circuit output<br>(Select this when the active high setting is specified (TAUDnTOL04 = 0).) |
| 1 | 1 | Inverted combination circuit output<br>(Select this when the active low setting is specified (TAUDnTOL04 = 1).) |
| Other than above | | Setting prohibited |

| | | |
|---|---|---|
| 15 to 0 | Reserved | *1 |

Note 1. Some of the bits defined as 0 in the PIC0REG2n1 register are defined for the other timer connection functions. For such bits, use the bit definition of those timer connection functions.

### 29.12.3.3 PIC0REG2n2 — Timer I/O Control Register 2n2 (n = 0)

**Access:** This register can be read or written in 32-bit units.

**Address:** PIC0REG202: FFDD 00C8$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | PIC0REG2n225 | — | — | — | PIC0REG2n221 | — | — | — | PIC0REG2n217 | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R | R | R | R/W | R | R | R | R/W | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 29.54 PIC0REG2n2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | *1 |
| 25 | PIC0REG2n225 | Select the TIN input signal to TAUDTTIN14. |

| PIC0REG2n225 | Input signal |
|---|---|
| 1 | Signal selected by the PIC0REG2n018 bit (TOUT of CH2 of TAUDn) |
| Other than above | Setting prohibited |

| Bit Position | Bit Name | Function |
|---|---|---|
| 24 to 22 | Reserved | *1 |
| 21 | PIC0REG2n221 | Select the TIN input signal to TAUDTTIN12. |

| PIC0REG2n221 | Input signal |
|---|---|
| 1 | Signal selected by the PIC0REG2n018 bit (TOUT of CH2 of TAUDn) |
| Other than above | Setting prohibited |

| Bit Position | Bit Name | Function |
|---|---|---|
| 20 to 18 | Reserved | *1 |
| 17 | PIC0REG2n217 | Select the TIN input signal to TAUDTTIN10. |

| PIC0REG2n217 | Input signal |
|---|---|
| 1 | Signal selected by the PIC0REG2n018 bit (TOUT of CH2 of TAUDn) |
| Other than above | Setting prohibited |

| Bit Position | Bit Name | Function |
|---|---|---|
| 16 to 0 | Reserved | *1 |

Note 1. Some of the bits defined as 0 in the PIC0REG2n2 register are defined for the other timer connection functions. For such bits, use the bit definition of those timer connection functions.

### 29.12.3.4 PIC0REG2n3 — Timer I/O Control Register 2n3 (n = 0)

This register selects the logic of a combination circuit.

**Access:** This register can be read or written in 32-bit units.

**Address:** PIC0REG203: FFDD 00CC$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | PIC0REG2n322 | PIC0REG2n321 | PIC0REG2n320 | — | PIC0REG2n318 | PIC0REG2n317 | PIC0REG2n316 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | PIC0REG2n314 | PIC0REG2n313 | PIC0REG2n312 | — | PIC0REG2n310 | PIC0REG2n309 | PIC0REG2n308 | — | PIC0REG2n306 | PIC0REG2n305 | PIC0REG2n304 | — | PIC0REG2n302 | PIC0REG2n301 | PIC0REG2n300 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

**Table 29.55 PIC0REG2n3 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | *1 |
| 22<br>21<br>20 | PIC0REG2n322<br>PIC0REG2n321<br>PIC0REG2n320 | Select the logical operation to perform on input signals A and B according to the output logic specified for CH9 of TAUDn. |

| PIC0REG2n322 | PIC0REG2n321 | PIC0REG2n320 | Input signal |
|---|---|---|---|
| 1 | 0 | 0 | A and B<br>(Select this when the active high setting is specified (TAUDnTOL09 = 0).) |
| 1 | 0 | 1 | A or B<br>(Select this when the active low setting is specified (TAUDnTOL09 = 1).) |
| Other than above | | | Setting prohibited |

| Bit Position | Bit Name | Function |
|---|---|---|
| 19 | Reserved | *1 |
| 18<br>17<br>16 | PIC0REG2n318<br>PIC0REG2n317<br>PIC0REG2n316 | Select the logical operation to perform on input signals A and B according to the output logic specified for CH8 of TAUDn. |

| PIC0REG2n318 | PIC0REG2n317 | PIC0REG2n316 | Input signal |
|---|---|---|---|
| 1 | 0 | 0 | A and B<br>(Select this when the active high setting is specified (TAUDnTOL08 = 0).) |
| 1 | 0 | 1 | A or B<br>(Select this when the active low setting is specified<br>(TAUDnTOL08 = 1).) |
| Other than above | | | Setting prohibited |

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | Reserved | *1 |

**Table 29.55 PIC0REG2n3 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 14<br>13<br>12 | PIC0REG2n314<br>PIC0REG2n313<br>PIC0REG2n312 | Select the logical operation to perform on input signals A and B according to the output logic specified for CH7 of TAUDn. |

| PIC0REG 2n314 | PIC0REG 2n313 | PIC0REG 2n312 | Input signal |
|---|---|---|---|
| 1 | 0 | 0 | A and B<br>(Select this when the active high setting is specified (TAUDnTOL07 = 0).) |
| 1 | 0 | 1 | A or B<br>(Select this when the active low setting is specified (TAUDnTOL07 = 1).) |
| Other than above | | | Setting prohibited |

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 | Reserved | *1 |
| 10<br>9<br>8 | PIC0REG2n310<br>PIC0REG2n309<br>PIC0REG2n308 | Select the logical operation to perform on input signals A and B according to the output logic specified for CH6 of TAUDn. |

| PIC0REG 2n310 | PIC0REG 2n309 | PIC0REG 2n308 | Input signal |
|---|---|---|---|
| 1 | 0 | 0 | A and B<br>(Select this when the active high setting is specified (TAUDnTOL06 = 0).) |
| 1 | 0 | 1 | A or B<br>(Select this when the active low setting is specified (TAUDnTOL06 = 1).) |
| Other than above | | | Setting prohibited |

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | *1 |
| 6<br>5<br>4 | PIC0REG2n306<br>PIC0REG2n305<br>PIC0REG2n304 | Select the logical operation to perform on input signals A and B according to the output logic specified for CH5 of TAUDn. |

| PIC0REG 2n306 | PIC0REG 2n305 | PIC0REG 2n304 | Input signal |
|---|---|---|---|
| 1 | 0 | 0 | A and B<br>(Select this when the active high setting is specified (TAUDnTOL05 = 0).) |
| 1 | 0 | 1 | A or B<br>(Select this when the active low setting is specified (TAUDnTOL05 = 1).) |
| Other than above | | | Setting prohibited |

| Bit Position | Bit Name | Function |
|---|---|---|
| 3 | Reserved | *1 |
| 2<br>1<br>0 | PIC0REG2n302<br>PIC0REG2n301<br>PIC0REG2n300 | Select the logical operation to perform on input signals A and B according to the output logic specified for CH4 of TAUDn. |

| PIC0REG 2n302 | PIC0REG 2n301 | PIC0REG 2n300 | Input signal |
|---|---|---|---|
| 1 | 0 | 0 | A and B<br>(Select this when the active high setting is specified (TAUDnTOL04 = 0).) |
| 1 | 0 | 1 | A or B<br>(Select this when the active low setting is specified (TAUDnTOL04 = 1).) |
| Other than above | | | Setting prohibited |

Note 1. Some of the bits defined as 0 in the PIC0REG2n3 register are defined for the other timer connection functions. For such bits, use the bit definition of those timer connection functions.

### 29.12.3.5 PIC0HIZCENn — Hi-Z Output Control Register n (n = 0)

This register selects the Hi-Z output control input signal of TAPAn.

**Access:** This register can be read or written in 8-bit units.

**Address:** PIC0HIZCEN0: FFDD $0080_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | PIC0HIZCENn6 | — | — | — | PIC0HIZCENn2 | — | PIC0HIZCENn0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R | R | R | R/W | R | R/W |

**Table 29.56 PIC0HIZCENn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 6 | PIC0HIZCENn6 | Select whether to enable or disable Hi-Z output control by the INTADCA0ERR interrupt signal.<br>0: Disable<br>1: Enable |
| 5 to 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 | PIC0HIZCENn2 | Select whether to enable or disable Hi-Z output control by the WDTA1NMI interrupt signal.<br>0: Disable<br>1: Enable |
| 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | PIC0HIZCENn0 | Select whether to enable or disable Hi-Z output control by the TAPAnESO pin input.<br>0: Disable<br>1: Enable |

### 29.12.4 Operation Example

This is achieved by combining the following TAUD features:

- Triangle PWM output with dead time
- One-shot pulse output

In addition, the following PIC is also used because the pulse to be inserted into the variable dead time area is generated for the positive or negative phase:

- Combination circuit (PFN001, PFN023, and PFN045)

In addition, the following peripheral interconnections are also used because the pulse to be inserted into the variable dead time area is combined with the triangle PWM output waveform:

- Logical operation circuit (FN0i) (i = 0 to 5)

A high-accuracy triangle PWM signal with dead time is output by assigning the PWM output achieved using the above features to the U, V, and W phases. Therefore, the PWM output dead time can be freely specified for the PWM signal of each phase. Because the only difference among phases is the assigned channel, only one phase (the U phase) is described below.

#### 29.12.4.1 Triangle PWM Output with Dead Time

A triangle PWM signal with dead time is output from TOUT04 and TOUT05 by using CH2, CH4, and CH5 in combination.

#### 29.12.4.2 One-shot Pulse Output

A CDR11 pulse for which the width is delayed by the delay time (CDR10) from the valid edge of the TIN10 (TOUT02) signal of CH10 is output as TOUT11 by using CH10 and CH11 in combination.

This pulse is used as the variable dead time area pulse used near duties of 100% and 0%.

**CAUTION**

**Specify each CDR setting for one-shot pulse output such that the following condition is satisfied: CDR05 ≥ (CDR10 + CDR11)**

**If a value that does not satisfy the above condition is specified, the output waveform might be affected. To minimize this effect, in addition to satisfying the above setting condition, leave CDR11 set to $0000_H$ until the variable dead time area pulse is required. Detect both rising and falling edges as the valid TIN10 (TOUT02) edge, and set TAUDnTOL11 to 1 (active low).**
**Specify the same operation clock for each TAUDn channel used for outputting a triangle PWM signal with dead time or a one-shot pulse.**

For details about the TAUD functions, see **Section 25, Timer Array Unit D (TAUD)**.

### 29.12.4.3 U phase Combination Circuit (PFN001)

This circuit generates a variable dead time area pulse (FN00 A, FN01 A) for adding a generated one-shot pulse to a generated triangle PWM signal with dead time.

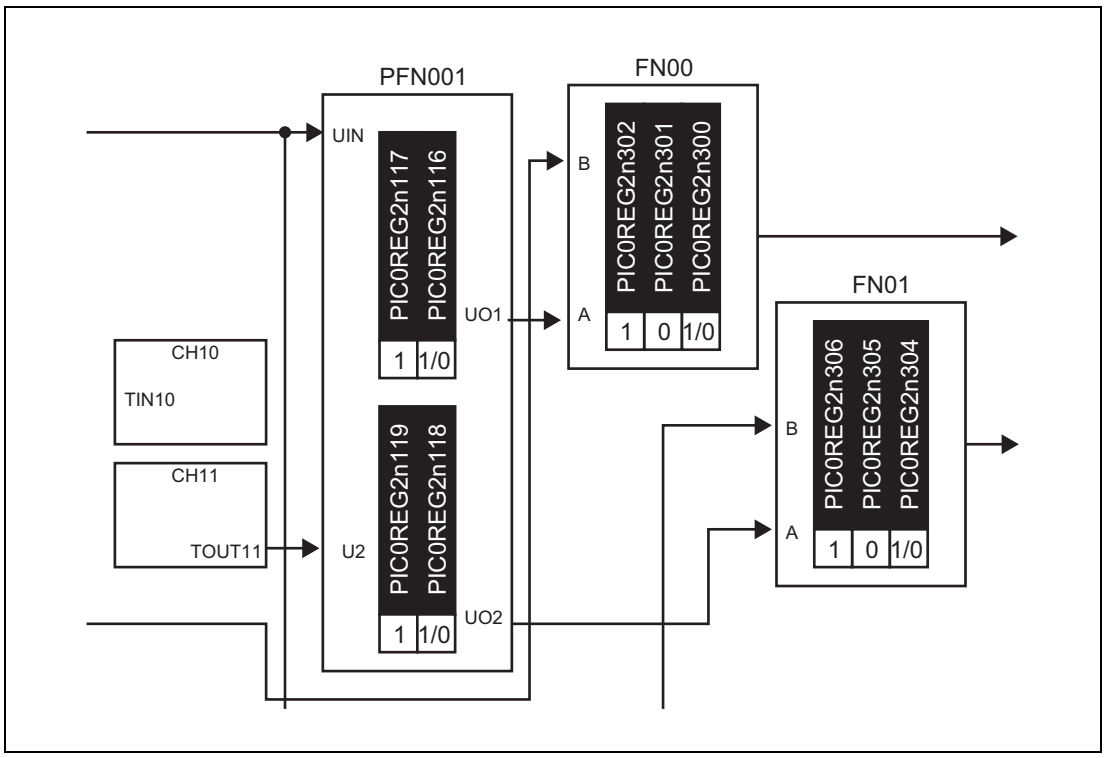



**Figure 29.35 Block Diagram Excerpt (PFN001, FN00, and FN01)**

The table below shows the relationships between combination circuit input (UIN, U2) and output (UO1, UO2).

**Table 29.57 U and UB Phase Combination Circuit (PFN001) I/O Table**

- **UO1 (U phase variable dead time area pulse) output**

| UIN (TOUT02) | U2 (TOUT11) | UO1 | |
|---|---|---|---|
| | | PIC0REG2n117, 16 = $10_B$ U phase output active high (TAUDnTOL04 = 0) | PIC0REG2n117, 16 = $11_B$ U phase output active low (TAUDnTOL04 = 1) |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

- **UO2 (UB phase variable dead time area pulse) output**

| UIN (TOUT02) | U2 (TOUT11) | UO2 | |
|---|---|---|---|
| | | PIC0REG2n119, 18 = $10_B$ UB phase output active high (TAUDnTOL05 = 0) | PIC0REG2n119, 18 = $11_B$ UB phase output active low (TAUDnTOL05 = 1) |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 |

**NOTE**

The PIC0REG2n116, PIC0REG2n117, PIC0REG2n118, and PIC0REG2n119 settings change depending on the active U phase and UB phase levels of the generated triangle PWM signal with dead time.

### 29.12.4.4 Logical Operation Circuit (FN0i) (i = 0 or 1)

This circuit combines an output triangle PWM signal with dead time (TOUT04, TOUT05) with combination circuit output (UO1 and UO2 of PFN001) and generates a PWM signal to which a variable dead time area pulse has been added.

The combination logic for the logical operation circuit is switched according to the PIC0REG2n3 register setting. (Bits 0 to 2 are specified for U phase output, and bits 4 to 6 are specified for UB phase output.)

Set up the logical operation circuit as shown in the table below. The combined signal is output from the TAPAnUP and TAPAnUM pins according to the specified combination logic.

**Table 29.58 Logical Operation Circuit (FN0i) (i = 0 or 1) Settings and TAPAnUP and TAPAnUM Pin Output**

- **U phase output (TOUT04)**

| Active level | PIC0REG2n302 to 00 | TAPAnUP pin output waveform |
|---|---|---|
| Active high (TAUDnTOL04 = 0) | $100_B$ | AND of FN00 B (TOUT04) and FN00 A (UO1) |
| Active low (TAUDnTOL04 = 1) | $101_B$ | OR of FN00 B (TOUT04) and FN00 A (UO1) |

- **UB phase output (TOUT05)**

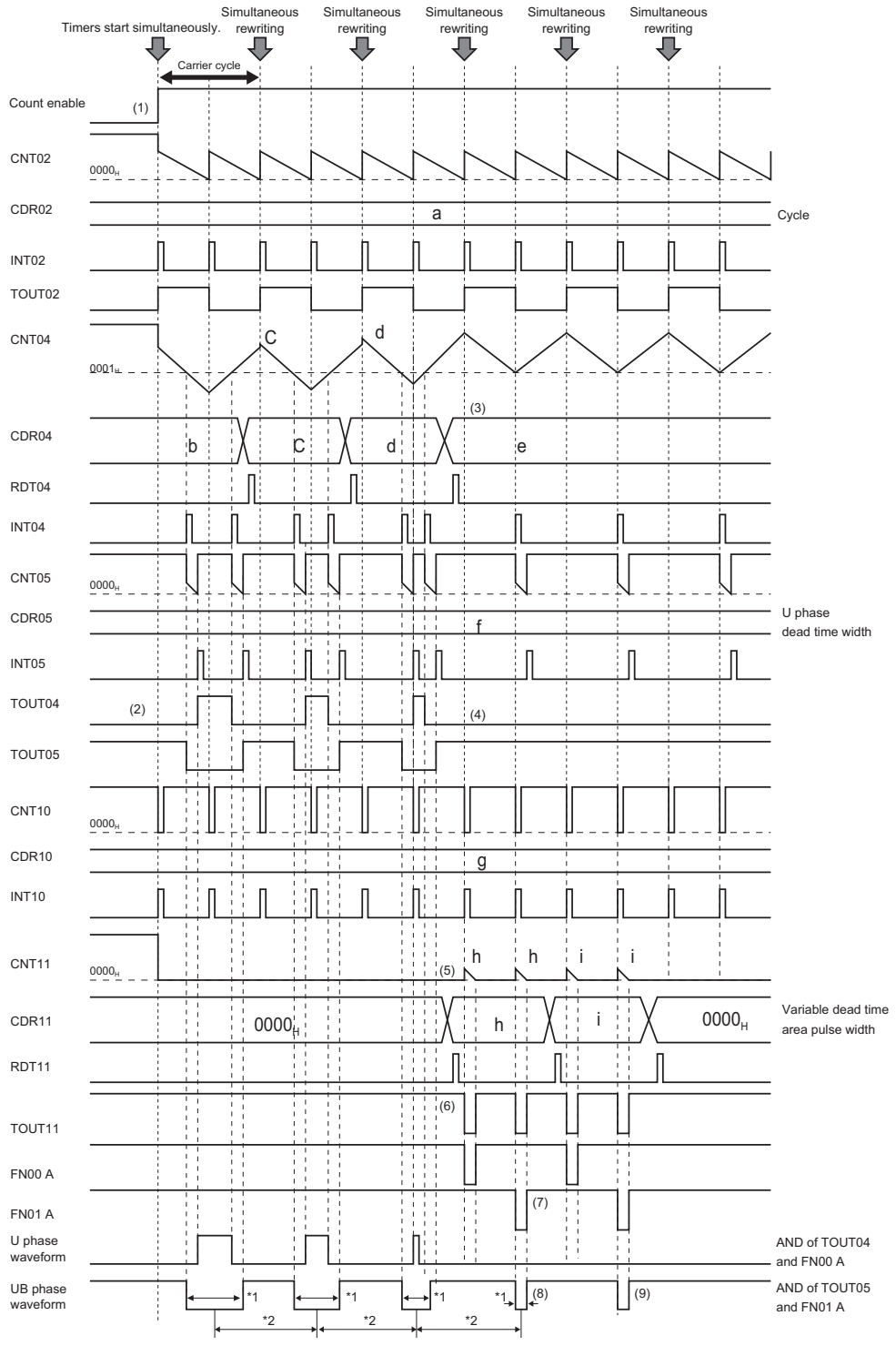
| Active level | PIC0REG2n306 to 04 | TAPAnUM pin output |
|---|---|---|
| Active high (TAUDnTOL05 = 0) | $100_B$ | AND of FN01 B (TOUT05) and FN01 A (UO2) |
| Active low (TAUDnTOL05 = 1) | $101_B$ | OR of FN01 B (TOUT05) and FN01 A (UO2) |

Because the above makes variable dead time control possible to ensure output accuracy near duties of 0% and 100% even for TAUD, a more accurate triangle PWM signal can be output than that output using the TAUD feature for outputting a triangle PWM signal with dead time.

For the V/VB phase and W/WB phase, the used channels and register bits differ, but the settings are the same, as shown in **Figure 29.34, Block Diagram: High-Accuracy Triangle PWM Output with Dead Time.**

The peripheral interconnections provide a connection for adding the pulse generated during one-shot pulse output to the PWM signal generated during output of a triangle PWM signal with dead time by using the combination circuit and logical operation circuit of the peripheral interconnections.

The following figures show timing charts for outputting a high-accuracy triangle PWM signal with dead time.




Note 1. The variable dead time area pulse uses a sawtooth wave and is therefore expanded on one side, unlike a pulse that uses a triangle wave, which is expanded on both sides.

Note 2. Because the variable dead time area pulse is expanded on one side, the length of the one-phase PWM output cycle for the variable dead time area increases by 1/2 the added variable dead time area pulse width.

**Figure 29.36 Example of a High-Accuracy PWM Signal Output with Dead Time (U Phase: 0%, UB Phase: 100%) (when TAUDnTOL04 = 0 (Active High) and TAUDnTOL05 = 0 (Active High))**

An operation example in which the system transitions to a U phase of 0% and UB phase of 100% in the timer configuration for performing the U phase PWM output shown in **Figure 29.36** is provided below. Output of a triangle PWM signal with dead time is active high.

(1) When timer operation is started, output of a triangle PWM with dead time is started by the CH2, CH4, and CH5 channels of TAUDn.

(2) A triangle PWM waveform with dead time is generated from TOUT04 and TOUT05.

(3) A U phase duty output value of 0% is specified for CDR04.

(4) Due to the setting in (3), TOUT04 output is the inactive level, and TOUT05 output is the active level. However, no variable dead time area pulse is output during this operation.

(5) To create a variable dead time area pulse, the value to be used as the pulse width is specified for CDR11 when specifying the 0% U phase duty in (3).
For this example, the CDR11 setting is fixed to $0000_H$ until the system enters the variable dead time area to prevent adverse effects on the output PWM signal.

(6) The variable dead time area pulse is output as a pulse that has the width specified for CDR11 after the delay time specified for CDR10 elapses, starting at the TOUT02 edge.

(7) The pulse output in (6) is converted to a variable dead time area pulse for the U phase (FN00 A) and UB phase (FN01 A) by the combination circuit (PFN001).

(8) The pulse generated in (7) is combined with the TOUT04 and TOUT05 output waveforms by using the logical operation circuits (FN00, FN01), and the result is output from TAPAnUP (U phase output) and TAPAnUM (UB phase output).

(9) By later changing the CDR11 setting, which specifies the width of the variable dead time area pulse, the desired variable dead time area pulse can be added.

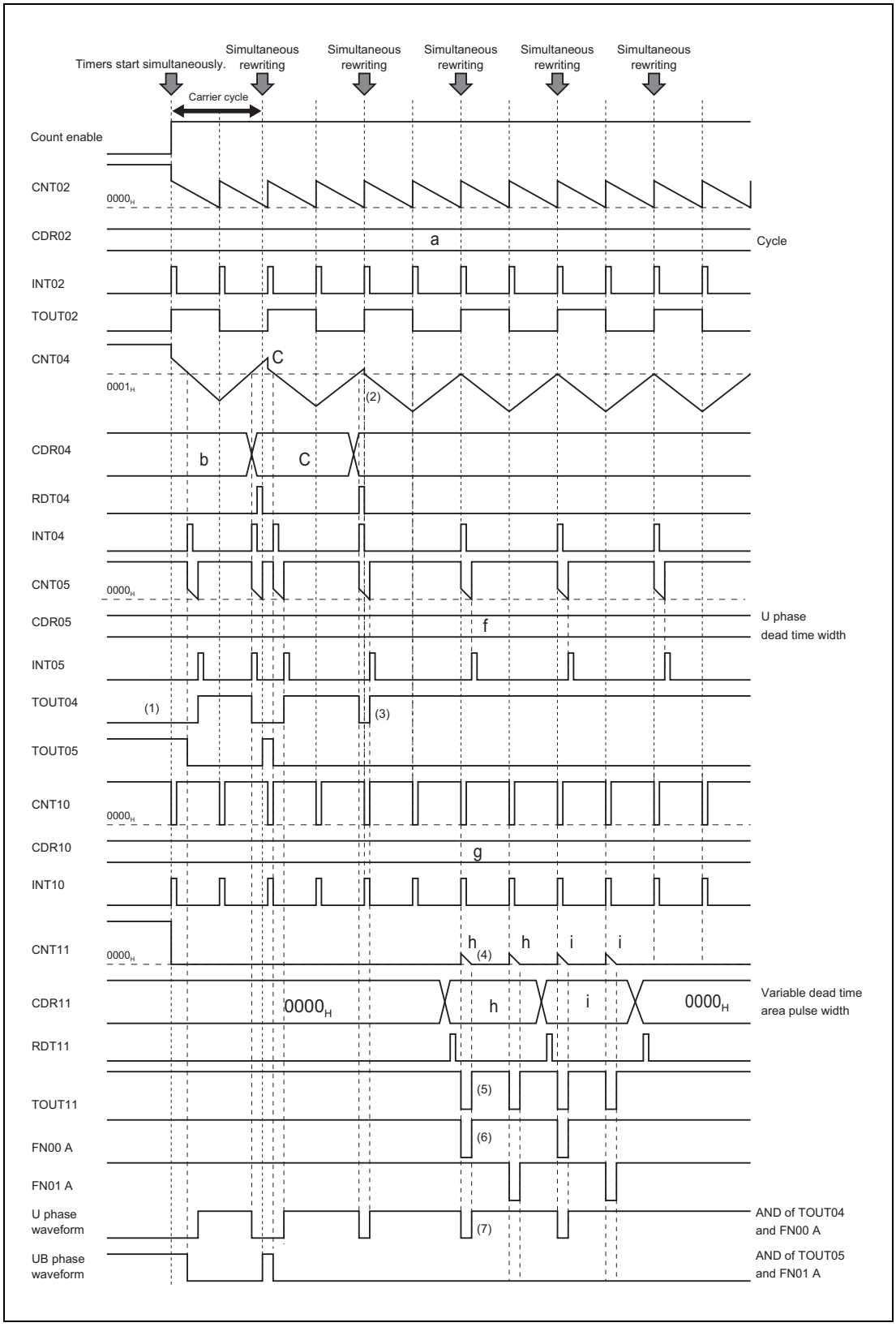



**Figure 29.37 Example of a High-Accuracy PWM Signal Output with Dead Time (U Phase: 100%, UB Phase: 0%) (when TAUDnTOL04 = 0 (Active High) and TAUDnTOL05 = 0 (Active High))**

An operation example in which the system transitions to a U phase of 100% and UB phase of 0% in the timer configuration for performing the U phase PWM output shown in **Figure 29.37** is provided below. Output of a triangle PWM signal with dead time is active high.

(1) The timer operation from the start of timer operation until the output of a triangle PWM signal with dead time is the same.

(2) A U phase duty output value of 100% (CDR04 = $0000_H$) is specified for CDR04.

(3) Due to the setting in (2), TOUT04 output is the active level, and TOUT05 output is the inactive level. However, no variable dead time area pulse is output during this operation.

(4) To create a variable dead time area pulse, the value to be used as the pulse width is specified for CDR11 one cycle after specifying the 100% U phase duty setting in (2).
For this example, the CDR11 setting is fixed to $0000_H$ until the system enters the variable dead time area to prevent adverse effects on the output PWM signal.

(5) The variable dead time area pulse is output as a pulse that has the width specified for CDR11 after the delay time specified for CDR10 elapses, starting at the TOUT02 edge.

(6) The pulse output in (5) is converted to a variable dead time area pulse for the U phase (FN00 A) and UB phase (FN01 A) by the combination circuit (PFN001).

(7) The pulse generated in (6) is combined with the TOUT04 and TOUT05 output waveforms by using the logical operation circuits (FN00, FN01), and the result is output from TAPAnUP (U phase output) and TAPAnUM (UB phase output).

**CAUTION**

**If the 100% U phase duty setting for CDR04 and the variable dead time area pulse width for CDR11 are specified at the same time, the variable dead time area pulse is affected by the amount shown by (2) for the last PWM signal output from TOUT04 and shown by feature specification (1), as shown in Figure 29.37.**
**To cancel this effect, the CDR11 setting is delayed one cycle.**

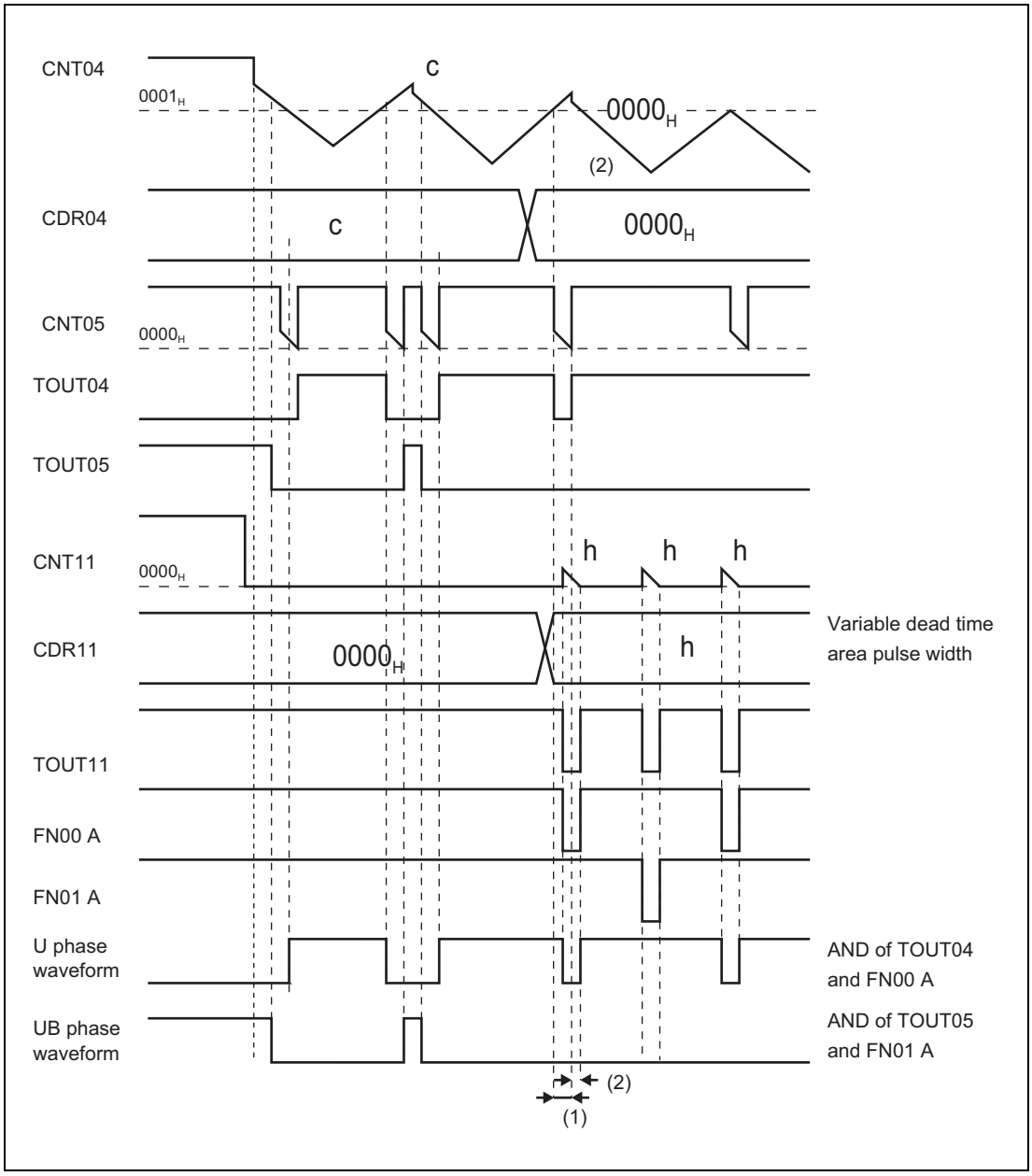



**Figure 29.38 Effect on the Output Triangle PWM Wave with Dead Time by the Variable Dead Time Area Pulse**

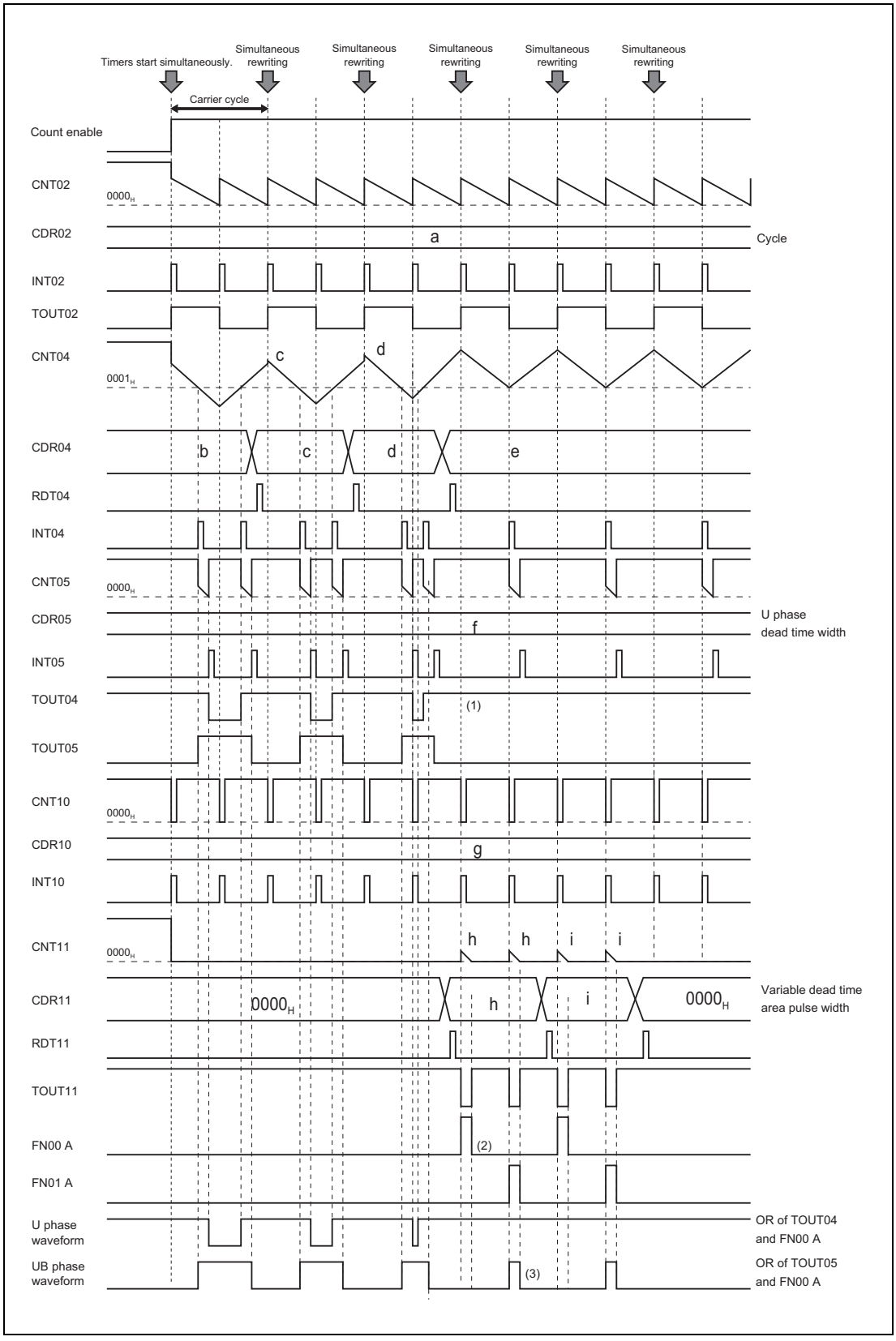



**Figure 29.39 Example of a High-Accuracy PWM Signal Output with Dead Time (U Phase: 100%, UB Phase: 0%) (TAUDnTOL04 = 1 (Active Low), TAUDnTOL05 = 1 (Active Low))**

An operation example in which the system transitions to a U phase of 100% and UB phase of 0% in the timer configuration for performing the U phase PWM output shown in **Figure 29.39** is provided below. Output of a triangle PWM signal with dead time is active low.

(1) The timer operation from the start of timer operation until the output of a triangle PWM with dead time is the same as in **Figure 29.36, Example of a High-Accuracy PWM Signal Output with Dead Time (U Phase: 0%, UB Phase: 100%) (when TAUDnTOL04 = 0 (Active High) and TAUDnTOL05 = 0 (Active High))**. However, an active low PWM signal is output from TOUT04 and TOUT05.

(2) Therefore, active low output that corresponds with PWM output is specified as the combination circuit setting (PIC0REG2n116 and PIC0REG2n117, and PIC0REG2n118 and PIC0REG2n119). This results in the output of an active low variable dead time area pulse for the U phase (FN00 A) and UB phase (FN01 A).

(3) In addition, active low output that corresponds with PWM output is also specified as the logical operation circuit setting (PIC0REG2n302 to PIC0REG2n300 and PIC0REG2n306 to PIC0REG2n304). The pulse generated in (2) is combined with the TOUT04 and TOUT05 output waveforms, and the result is output from TAPAnUP (U phase output) and TAPAnUM (UB phase output) as an active low PWM signal.

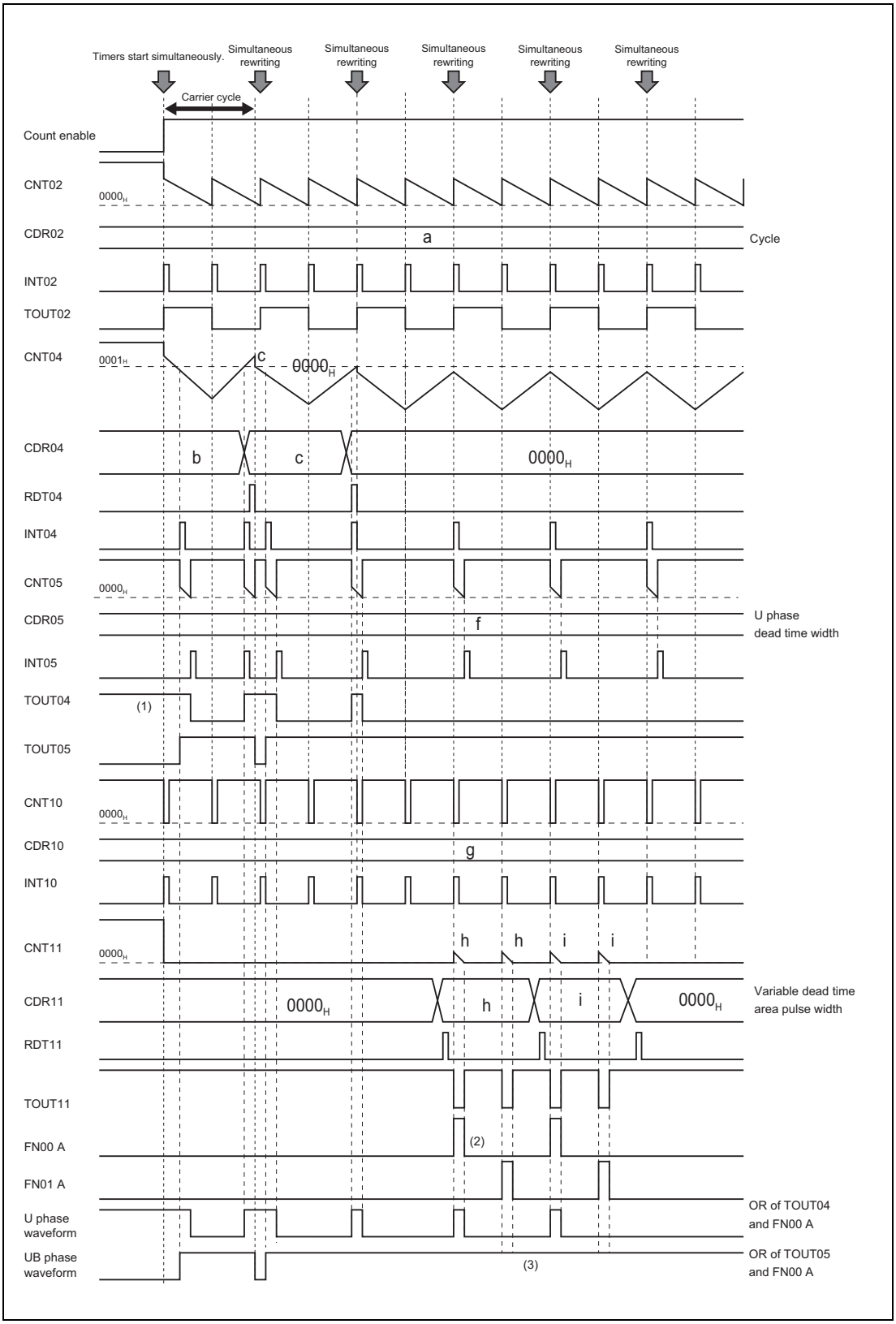



**Figure 29.40 Example of a High-Accuracy PWM Signal Output with Dead Time (U Phase: 0%, UB Phase: 100%) (when TAUDnTOL04 = 0 (Active Low) and TAUDnTOL05 = 0 (Active Low))**

An operation example in which the system transitions to a U phase of 0% and UB phase of 100% in the timer configuration for performing the U phase PWM output shown in **Figure 29.40** is provided below. Output of a triangle PWM signal with dead time is active low.

(1) The timer operation from the start of timer operation until the output of a triangle PWM signal with dead time is the same as in **Figure 29.37, Example of a High-Accuracy PWM Signal Output with Dead Time (U Phase: 100%, UB Phase: 0%) (when TAUDnTOL04 = 0 (Active High) and TAUDnTOL05 = 0 (Active High))**. However, an active low PWM signal is output.

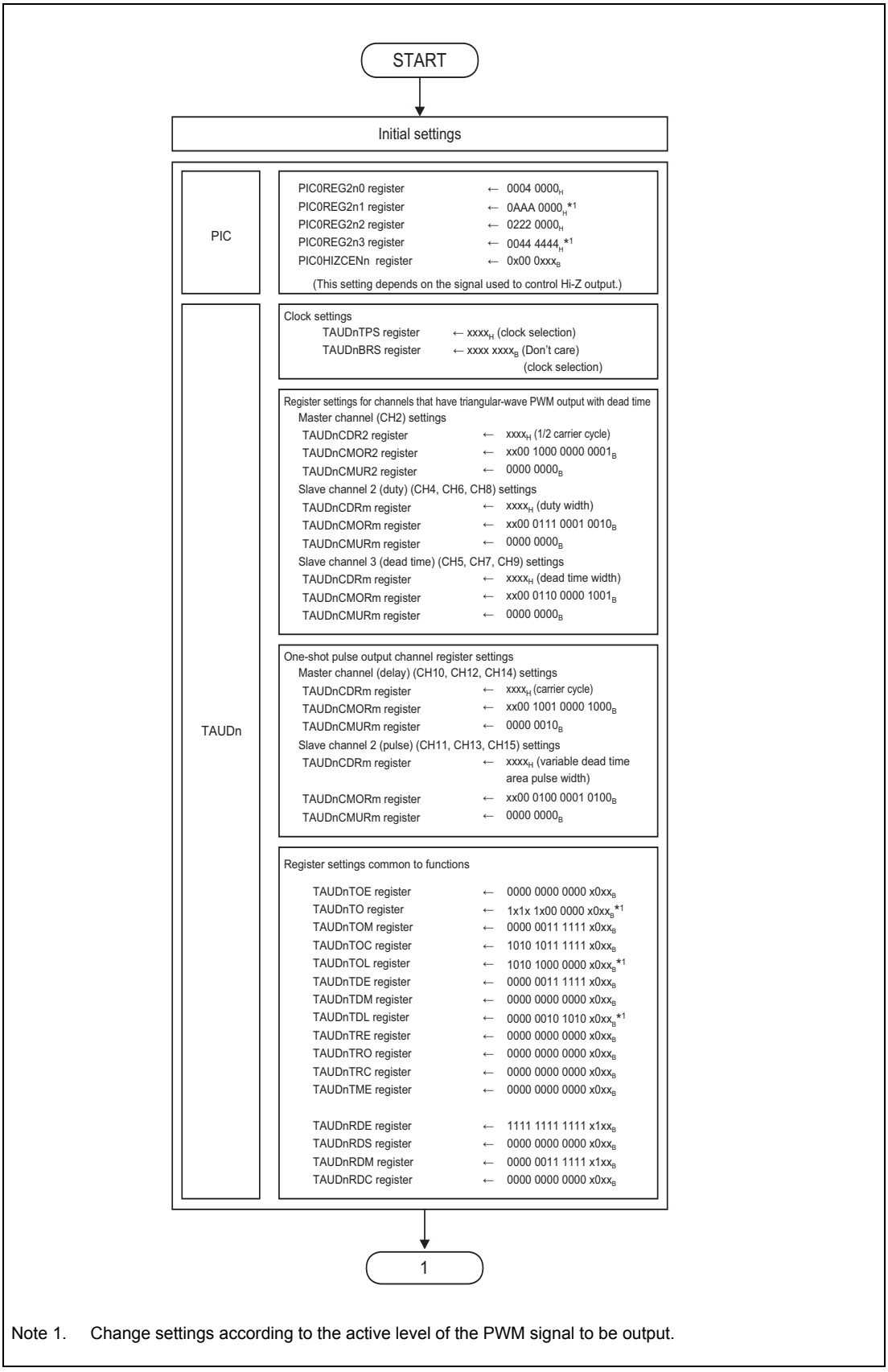
(2) Therefore, active low output that corresponds with PWM output is specified as the combination circuit setting (PIC0REG2n116 and PIC0REG2n117, and PIC0REG2n118 and PIC0REG2n119). This results in the output of an active low variable dead time area pulse for the U phase (FN00 A) and UB phase (FN01 A).

(3) In addition, active low output that corresponds with PWM output is also specified as the logical operation circuit setting (PIC0REG2n302 to PIC0REG2n300 and PIC0REG2n306 to PIC0REG2n304). The pulse generated in (2) is combined with the TOUT04 and TOUT05 output waveforms, and the result is output from TAPAnUP (U phase output) and TAPAnUM (UB phase output) as an active low PWM signal.

**CAUTION**

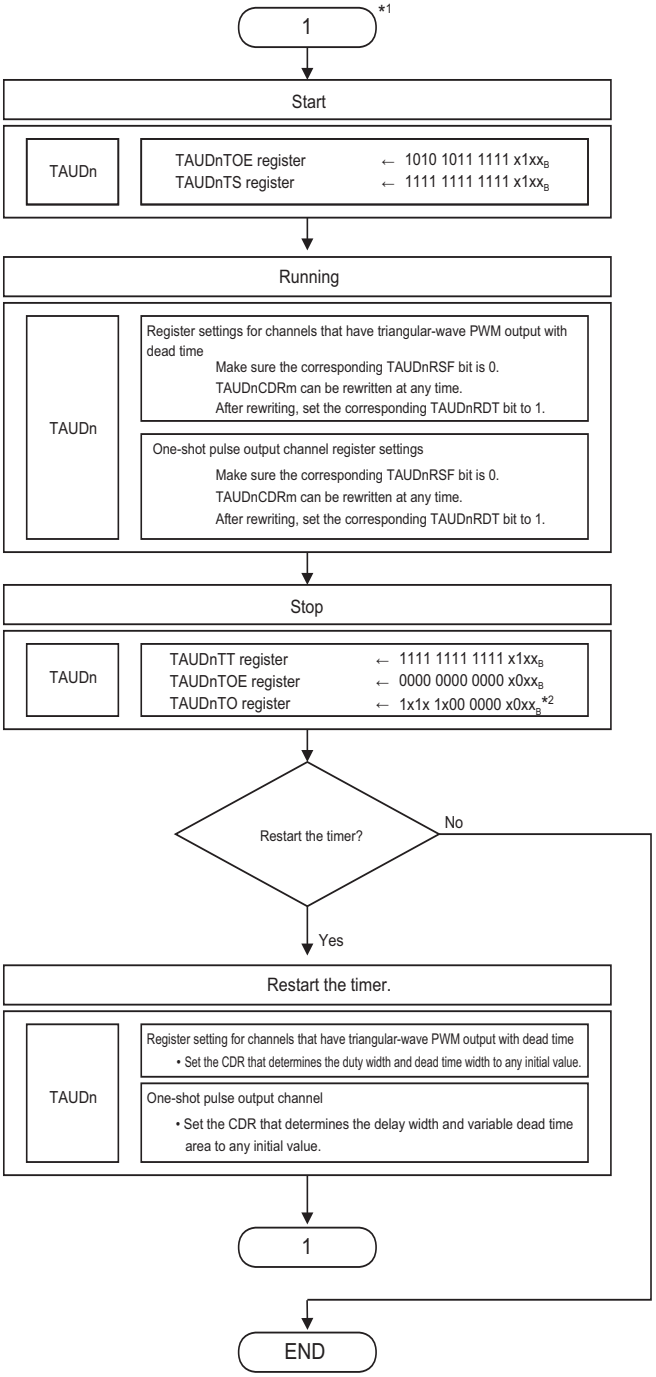
**If the 100% U phase duty setting for CDR04 and the variable dead time area pulse width for CDR11 are specified at the same time, the last PWM signal output from TOUT04 is adversely affected due to the feature specifications.**
**To cancel this effect, the CDR11 setting is delayed one cycle.**
**For details, see Figure 29.38, Effect on the Output Triangle PWM Wave with Dead Time by the Variable Dead Time Area Pulse.**

### 29.12.5 Setup Flow




Note 1. Change settings according to the active level of the PWM signal to be output.

**Figure 29.41 Setup Flow (Active High Example)**




Note 1. Specify the selection register and output port to use after specifying the initial settings for the peripheral interconnections and timers.

Note 2. Change settings according to the active level of the PWM signal to be output.

**Figure 29.42 Setup Flow (Active High Example) (continued)**

## 29.12.6 Setting Examples for Operation Functions

This section provides example settings for each register.

### 29.12.6.1 TAUDn settings (active high example)

**Table 29.59 TAUDn: CH2-related (Master Channel Used To Output A Triangle PWM Signal with Dead Time*1)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnCMOR2 | 15, 14 | TAUDnCKS[1:0] | Don't care*2 | Operation clock setting |
| | 13, 12 | TAUDnCCS[1:0] | 00 | |
| | 11 | TAUDnMAS | 1 | |
| | 10 to 8 | TAUDnSTS[2:0] | 000 | |
| | 7, 6 | TAUDnCOS[1:0] | 00 | |
| | 5 | | 0 | |
| | 4 to 1 | TAUDnMD[4:1] | 0000 | |
| | 0 | TAUDnMD0 | 1 | At the start of operation, output INTm and toggle TOUTm. |
| TAUDnCMUR2 | 1, 0 | TAUDnTIS[1:0] | 00 | Fixed |

Note 1. The master channel and slave channel names are defined for TAUD triangle PWM output with dead time. For details, see **Section 25, Timer Array Unit D (TAUD)**.

Note 2. The same operation clock must be specified for the master channel and slave channel.

**NOTE**

For the TAUDnCMORm register of the master channel used when outputting a triangle PWM signal with dead time, TAUDnCKS[1:0] (which selects the operation clock) and TAUDnMD0 can be set to any value, but other control bits have fixed values. For details, see **Section 25, Timer Array Unit D (TAUD)**.
For this feature, set TAUDnMD0 to 1.

**Table 29.60 TAUDn: CH4, CH6, and CH8-related (Slave Channel 2 used to Output a Triangle PWM Signal with Dead Time*1) (m = 4, 6, or 8)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnCMORm | 15, 14 | TAUDnCKS[1:0] | Don't care*2 | Operation clock setting |
| | 13, 12 | TAUDnCCS[1:0] | 00 | |
| | 11 | TAUDnMAS | 0 | |
| | 10 to 8 | TAUDnSTS[2:0] | 111 | |
| | 7, 6 | TAUDnCOS[1:0] | 00 | |
| | 5 | | 0 | |
| | 4 to 1 | TAUDnMD[4:1] | 1001 | |
| | 0 | TAUDnMD0 | 0 | |
| TAUDnCMURm | 1, 0 | TAUDnTIS[1:0] | 00 | |

Note 1. The same operation clock must be specified for the master channel and slave channel.
For the TAUDnCMORm register of slave channels 2 and 3, which is used when outputting a triangle PWM signal with dead time, TAUDnCKS[1:0] (which selects the operation clock) can be set to any value, but other control bits have fixed values. For details, see **Section 25, Timer Array Unit D (TAUD)**.

Note 2. The same operation clock must be specified for the master channel and slave channel.

**Table 29.61 TAUDn: CH5, CH7, and CH9-related (Slave Channel 3 used to Output a Triangle PWM Signal with Dead Time*1) (m = 5, 7, or 9)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnCMORm | 15, 14 | TAUDnCKS[1:0] | Don't care*2 | Operation clock setting |
| | 13, 12 | TAUDnCCS[1:0] | 00 | |
| | 11 | TAUDnMAS | 0 | |
| | 10 to 8 | TAUDnSTS[2:0] | 110 | |
| | 7, 6 | TAUDnCOS[1:0] | 00 | |
| | 5 | | 0 | |
| | 4 to 1 | TAUDnMD[4:1] | 0100 | |
| | 0 | TAUDnMD0 | 1 | |
| TAUDnCMURm | 1, 0 | TAUDnTIS[1:0] | 00 | |

Note 1. The same operation clock must be specified for the master channel and slave channel.
For the TAUDnCMORm register of slave channels 2 and 3, which is used when outputting a triangle PWM signal with dead time, TAUDnCKS[1:0] (which selects the operation clock) can be set to any value, but other control bits have fixed values. For details, see **Section 25, Timer Array Unit D (TAUD)**.

Note 2. The same operation clock must be specified for the master channel and slave channel.

**Table 29.62 TAUDn: CH10, CH12, and CH14-related (Master Channel used to Output a One-shot Pulse*1) (m = 10, 12, or 14)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnCMORm | 15, 14 | TAUDnCKS[1:0] | Don't care*2 | Operation clock setting |
| | 13, 12 | TAUDnCCS[1:0] | 00 | |
| | 11 | TAUDnMAS | 1 | |
| | 10 to 8 | TAUDnSTS[2:0] | 001 | |
| | 7, 6 | TAUDnCOS[1:0] | 00 | |
| | 5 | | 0 | |
| | 4 to 1 | TAUDnMD[4:1] | 0100 | |
| | 0 | TAUDnMD0 | 0 | Disable start triggers during counting. |
| TAUDnCMURm | 1, 0 | TAUDnTIS[1:0] | 10 | Detect both rising and falling edges as valid. |

Note 1. The master channel and slave channel names are defined for TAUD one-shot pulse output. For details, see **Section 25, Timer Array Unit D (TAUD)**.

Note 2. The same operation clock must be specified for the master channel and slave channel.

**Table 29.63 TAUDn: CH11, CH13, and CH15-related (Slave Channel used to Output a One-Shot Pulse*1) (m = 11, 13, or 15)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnCMORm | 15, 14 | TAUDnCKS[1:0] | Don't care*2 | Operation clock setting |
| | 13, 12 | TAUDnCCS[1:0] | 00 | |
| | 11 | TAUDnMAS | 0 | |
| | 10 to 8 | TAUDnSTS[2:0] | 100 | |
| | 7, 6 | TAUDnCOS[1:0] | 00 | |
| | 5 | | 0 | |
| | 4 to 1 | TAUDnMD[4:1] | 1010 | |
| | 0 | TAUDnMD0 | 0 | Disable start triggers during counting. |
| TAUDnCMURm | 1, 0 | TAUDnTIS[1:0] | 00 | |

Note 1. The master channel and slave channel names are defined for TAUD one-shot pulse output. For details, see **Section 25, Timer Array Unit D (TAUD)**.

Note 2. The same operation clock must be specified for the master channel and slave channel. Specify the same clock setting as for the master channel (CH2) used to output a triangle PWM signal with dead time.

**NOTE**

For the TAUDnCMORm register used during one-shot pulse output, TAUDnCKS[1:0] (which selects the operation clock) and TAUDnMD0 can be set to any value, but other control bits have fixed values. For details, see **Section 25, Timer Array Unit D (TAUD)**.
For this feature clear TAUDnMD0 to 0.

**Table 29.64 Common TAUDn Channel Settings (1/4)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnTOE | 15 | TAUDnTOE15 | 0<br>1 | Disable the timer.<br>Enable the timer. |
| | 14 | TAUDnTOE14 | 0 | |
| | 13 | TAUDnTOE13 | 0<br>1 | Disable the timer.<br>Enable the timer. |
| | 12 | TAUDnTOE12 | 0 | |
| | 11 | TAUDnTOE11 | 0<br>1 | Disable the timer.<br>Enable the timer. |
| | 10 | TAUDnTOE10 | 0 | |
| | 9 to 4 | TAUDnTOE09 to TAUDnTOE04 | 0<br>1 | Disable the timer.<br>Enable the timer. |
| | 3 | TAUDnTOE03 | Don't care | |
| | 2 | TAUDnTOE02 | 0<br>1 | Disable the timer.<br>Enable the timer. |
| | 1, 0 | TAUDnTOE01<br>TAUDnTOE00 | Don't care | |

**Table 29.64 Common TAUDn Channel Settings (2/4)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnTO | 15 | TAUDnTO15 | 1*1 | Output a high-level signal to TOUT15. |
| | 14 | TAUDnTO14 | Don't care | |
| | 13 | TAUDnTO13 | 1*1 | Output a high-level signal to TOUT13. |
| | 12 | TAUDnTO12 | Don't care | |
| | 11 | TAUDnTO11 | 1*1 | Output a high-level signal to TOUT11. |
| | 10 | TAUDnTO10 | Don't care | |
| | 9 to 4 | TAUDnTO09 to TAUDnTO04 | 0*1 | Output a low-level signal to TOUT09 to TOUT04. |
| | 3 | TAUDnTO03 | Don't care | |
| | 2 | TAUDnTO02 | 0 | Output a low-level signal to TOUT02. |
| | 1, 0 | TAUDnTO01 TAUDnTO00 | Don't care | |
| TAUDnTOM | 15 to 10 | TAUDnTOM15 to TAUDnTOM10 | 0 | Independent operation mode |
| | 9 to 4 | TAUDnTOM09 to TAUDnTOM04 | 1 | Synchronous operation mode |
| | 3 | TAUDnTOM03 | Don't care | |
| | 2 | TAUDnTOM02 | 0 | Independent operation mode |
| | 1, 0 | TAUDnTOM01 TAUDnTOM00 | Don't care | |
| TAUDnTOC | 15 | TAUDnTOC15 | 1 | Operation mode 2 |
| | 14 | TAUDnTOC14 | 0 | Operation mode 1 |
| | 13 | TAUDnTOC13 | 1 | Operation mode 2 |
| | 12 | TAUDnTOC12 | 0 | Operation mode 1 |
| | 11 | TAUDnTOC11 | 1 | Operation mode 2 |
| | 10 | TAUDnTOC10 | 0 | Operation mode 1 |
| | 9 to 4 | TAUDnTOC09 to TAUDnTOC04 | 1 | Operation mode 2 |
| | 3 | TAUDnTOC03 | Don't care | |
| | 2 | TAUDnTOC02 | 0 | Operation mode 1 |
| | 1, 0 | TAUDnTOC01 TAUDnTOC00 | Don't care | |
| TAUDnTOL | 15 | TAUDnTOL15 | 1*1 | Inverted logic output (active low) |
| | 14 | TAUDnTOL14 | Don't care | |
| | 13 | TAUDnTOL13 | 1*1 | Inverted logic output (active low) |
| | 12 | TAUDnTOL12 | Don't care | |
| | 11 | TAUDnTOL11 | 1*1 | Inverted logic output (active low) |
| | 10 | TAUDnTOL10 | Don't care | |
| | 9 to 4 | TAUDnTOL09 to TAUDnTOL04 | 0*1 | Positive logic output (active high) |
| | 3 | TAUDnTOL03 | Don't care | |
| | 2 | TAUDnTOL02 | 0 | Positive logic output (active high) |
| | 1, 0 | TAUDnTOL01 TAUDnTOL00 | Don't care | |

**Table 29.64 Common TAUDn Channel Settings (3/4)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnTDE | 15 to 10 | TAUDnTDE15 to TAUDnTDE10 | 0 | Disable dead time control. |
| | 9 to 4 | TAUDnTDE09 to TAUDnTDE04 | 1 | Enable dead time control.*2 |
| | 3 | TAUDnTDE03 | Don't care | |
| | 2 | TAUDnTDE02 | 0 | Disable dead time control. |
| | 1, 0 | TAUDnTDE01 TAUDnTDE00 | Don't care | |
| TAUDnTDM | 15 to 9 | TAUDnTDM15 to TAUDnTDM09 | 0 | |
| | 3 | TAUDnTDM03 | Don't care | |
| | 2 | TAUDnTDM02 | 0 | Invalid because dead time control is disabled. |
| | 1, 0 | TAUDnTDM01 TAUDnTDM00 | Don't care | |
| TAUDnTDL | 15 to 10 | TAUDnTDL15 to TAUDnTDL10 | 0 | Invalid because dead time control is disabled. |
| | 9 | TAUDnTDL09 | 1*1 | Dead time is in the negative segment of the W phase output |
| | 8 | TAUDnTDL08 | 0*1 | Dead time is in the positive segment of the W phase output |
| | 7 | TAUDnTDL07 | 1*1 | Dead time is in the negative segment of the V phase output |
| | 6 | TAUDnTDL06 | 0*1 | Dead time is in the positive segment of the V phase output |
| | 5 | TAUDnTDL05 | 1*1 | Dead time is in the negative segment of the U phase output |
| | 4 | TAUDnTDL04 | 0*1 | Dead time is in the positive segment of the U phase output |
| | 3 | TAUDnTDL03 | Don't care | |
| | 2 | TAUDnTDL02 | 0 | Invalid because dead time control is disabled. |
| | 1, 0 | TAUDnTDL01 TAUDnTDL00 | Don't care | |
| TAUDnTRE | 15 to 4 | TAUDnTRE15 to TAUDnTRE04 | 0 | Disable real-time output. |
| | 3 | TAUDnTRE03 | Don't care | |
| | 2 | TAUDnTRE02 | 0 | Disable real-time output. |
| | 1, 0 | TAUDnTRE01 TAUDnTRE00 | Don't care | |
| TAUDnTRO | 15 to 4 | TAUDnTRO15 to TAUDnTRO04 | 0 | Invalid because real-time output is disabled. |
| | 3 | TAUDnTRO03 | Don't care | |
| | 2 | TAUDnTRO02 | 0 | Invalid because real-time output is disabled. |
| | 1, 0 | TAUDnTRO01 TAUDnTRO00 | Don't care | |
| TAUDnTRC | 15 to 4 | TAUDnTRC15 to TAUDnTRC04 | 0 | Do not use this channel to generate the real-time output trigger. |
| | 3 | TAUDnTRC03 | Don't care | |
| | 2 | TAUDnTRC02 | 0 | Do not use this channel to generate the real-time output trigger. |
| | 1, 0 | TAUDnTRC01 TAUDnTRC00 | Don't care | |

**Table 29.64 Common TAUDn Channel Settings (4/4)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnTME | 15 to 4 | TAUDnTME15 to TAUDnTME04 | 0 | Disable modulation output for timer output and real-time output. |
| | 3 | TAUDnTME03 | Don't care | |
| | 2 | TAUDnTME02 | 0 | Disable modulation output for timer output and real-time output. |
| | 1, 0 | TAUDnTME01 TAUDnTME00 | Don't care | |
| TAUDnRDE | 15 to 4 | TAUDnRDE15 to TAUDnRDE04 | 1 | Enable simultaneous rewriting. |
| | 3 | TAUDnRDE03 | Don't care | |
| | 2 | TAUDnRDE02 | 1 | Enable simultaneous rewriting. |
| | 1, 0 | TAUDnRDE01 TAUDnRDE00 | Don't care | |
| TAUDnRDS | 15 to 4 | TAUDnRDS15 to TAUDnRDS04 | 0 | Do not enable simultaneous rewriting by using another upper channel. |
| | 3 | TAUDnRDS03 | Don't care | |
| | 2 | TAUDnRDS02 | 0 | Do not enable simultaneous rewriting by using another upper channel. |
| | 1, 0 | TAUDnRDS01 TAUDnRDS00 | Don't care | |
| TAUDnRDM | 15 to 10 | TAUDnRDM15 to TAUDnRDM10 | 0 | Perform simultaneous rewriting when the master channel starts counting. |
| | 9 to 4 | TAUDnRDM09 to TAUDnRDM04 | 1 | Perform simultaneous rewriting after the master channel starts counting when there is a peak in the triangle wave on the corresponding slave channel. |
| | 3 | TAUDnRDM03 | Don't care | |
| | 2 | TAUDnRDM02 | 1 | Perform simultaneous rewriting after the master channel starts counting when there is a peak in the triangle wave on the corresponding slave channel. |
| | 1, 0 | TAUDnRDM01 TAUDnRDM00 | Don't care | |
| TAUDnRDC | 15 to 4 | TAUDnRDC15 to TAUDnRDC04 | 0 | Do not use this channel to generate the simultaneous rewrite trigger. |
| | 3 | TAUDnRDC03 | Don't care | |
| | 2 | TAUDnRDC02 | 0 | Do not use this channel to generate the simultaneous rewrite trigger. |
| | 1, 0 | TAUDnRDC01 TAUDnRDC00 | Don't care | |

Note 1. Change the setting according to the used system.

Note 2. These are used to control positive/negative phase waveform output for which even channels are paired with odd channels to perform dead time control. For details, see **Section 25, Timer Array Unit D (TAUD)**.

### 29.12.6.2 PIC Settings (Active High Example)

Table 29.65 PIC Settings

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| PIC0REG2n0 | 18 | PIC0REG2n018 | 1 | Select the TOUT signal of CH2 of TAUDn. |
| PIC0REG2n1 | 27, 26 | PIC0REG2n127<br>PIC0REG2n126 | 1<br>0 | Negative W phase active high combination circuit output |
| | 25, 24 | PIC0REG2n125<br>PIC0REG2n124 | 1<br>0 | Positive W phase active high combination circuit output |
| | 23, 22 | PIC0REG2n123<br>PIC0REG2n122 | 1<br>0 | Negative V phase active high combination circuit output |
| | 21, 20 | PIC0REG2n121<br>PIC0REG2n120 | 1<br>0 | Positive V phase active high combination circuit output |
| | 19, 18 | PIC0REG2n119<br>PIC0REG2n118 | 1<br>0 | Negative U phase active high combination circuit output |
| | 17, 16 | PIC0REG2n117<br>PIC0REG2n116 | 1<br>0 | Positive U phase active high combination circuit output |
| PIC0REG2n2 | 25 | PIC0REG2n225 | 1 | Select the input selected by the PIC0REG2n018 bit. |
| | 21 | PIC0REG2n221 | 1 | Select the input selected by the PIC0REG2n018 bit. |
| | 17 | PIC0REG2n217 | 1 | Select the input selected by the PIC0REG2n018 bit. |
| PIC0REG2n3 | 22, 21, 20 | PIC0REG2n322<br>PIC0REG2n321<br>PIC0REG2n320 | 1<br>0<br>0 | Negative W phase active high logical operation circuit output |
| | 18, 17, 16 | PIC0REG2n318<br>PIC0REG2n317<br>PIC0REG2n316 | 1<br>0<br>0 | Positive W phase active high logical operation circuit output |
| | 14, 13, 12 | PIC0REG2n314<br>PIC0REG2n313<br>PIC0REG2n312 | 1<br>0<br>0 | Negative V phase active high logical operation circuit output |
| | 10, 9, 8 | PIC0REG2n310<br>PIC0REG2n309<br>PIC0REG2n308 | 1<br>0<br>0 | Positive V phase active high logical operation circuit output |
| | 6, 5, 4 | PIC0REG2n306<br>PIC0REG2n305<br>PIC0REG2n304 | 1<br>0<br>0 | Negative U phase active high logical operation circuit output |
| | 2, 1, 0 | PIC0REG2n302<br>PIC0REG2n301<br>PIC0REG2n300 | 1<br>0<br>0 | Positive U phase active high logical operation circuit output |

# 29.13 Delay Pulse Output with Dead Time

## 29.13.1 Functional Overview

This feature outputs a three-phase PWM with dead time that is later than the cycle timing by an amount equal to the delay amount.

Unlike the function of three-phase PWM output with dead time, a PWM signal that has a reset in the next cycle can be output.

## 29.13.2 Configuration

The unit and channel configuration for this feature is shown below. (n = 0, m = 0 to 15)

**Table 29.66 Configuration of Delay Pulse Output with Dead Time**

| Timer | Timer motor control function |
|---|---|
| TAUD0 CH2 to CH15 (used channels fixed) | TAPA0 |

The signal names used in the descriptions below are abbreviations. The actual signal names corresponding to each abbreviation are as follows:

- INTm → INTTAUDnIm (TAUDn channel m interrupt)
- TINm → TAUDTTINm (TAUDn channel m input)
- TOUTm → TAUDTTOUTm (TAUDn channel m output)
- CDRm → TAUDnCDRm (TAUDn channel m data register)
- CNTm → TAUDnCNTm (TAUDn channel m counter register)

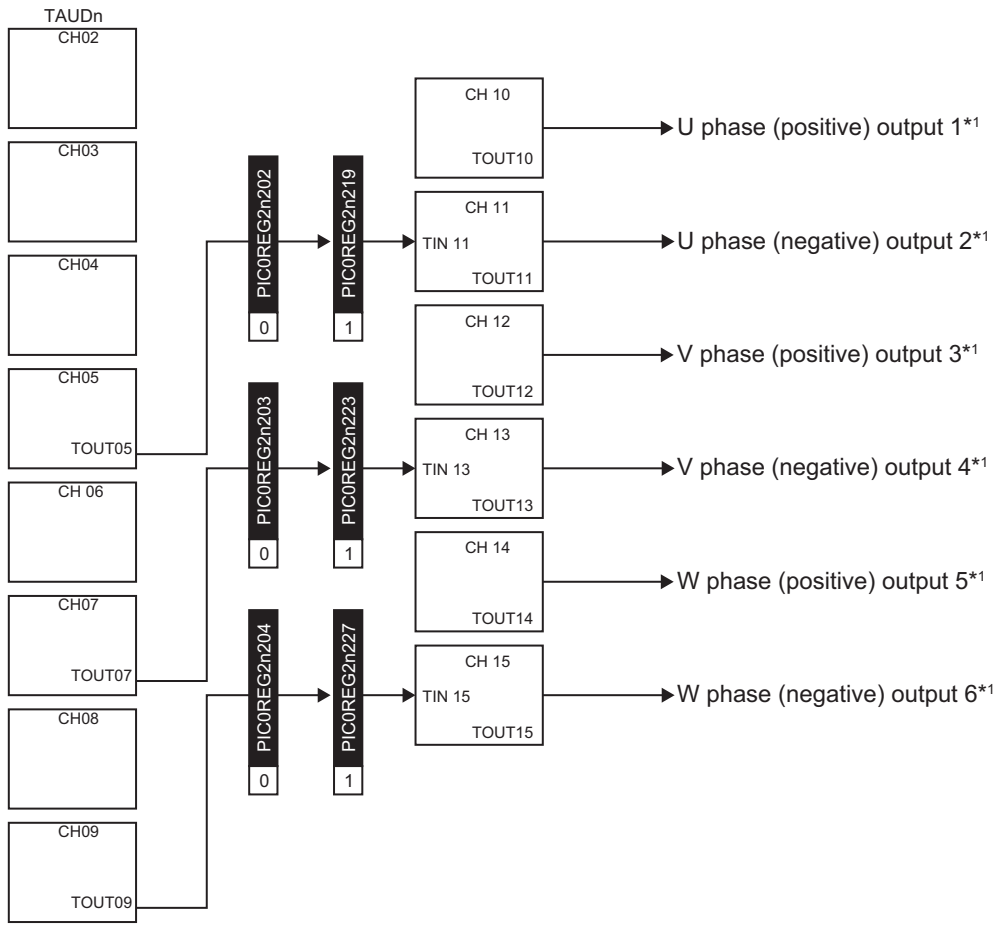
### 29.13.2.1 TAUDn configuration

Because the CDRm value of CH3 does not affect TOUT0 to TOUT15, the INTm signal of CH3 can also be used for other purposes such as A/D conversion trigger generation.

**Table 29.67 TAUDn configuration**

| CH | Function name | M/S*1 | CDR setting | Description |
|---|---|---|---|---|
| 2 | Delay pulse output function (CH2 is the master channel for CH3 to CH9.) | M | Cycle | |
| 3 | | S | | Reserved |
| 4 | | S | Delay (U phase) | |
| 5 | | S | Pulse width (U phase) | |
| 6 | | S | Delay (V phase) | |
| 7 | | S | Pulse width (V phase) | |
| 8 | | S | Delay (W phase) | |
| 9 | | S | Pulse width (W phase) | |
| 10 | Any feature that does not use TOUTm | S | | TOUT: U phase output |
| 11 | One-phase PWM output | S | Dead time (U phase) | TOUT: UB phase output |
| 12 | Any feature that does not use TOUTm | S | | TOUT: V phase output |
| 13 | One-phase PWM output | S | Dead time (V phase) | TOUT: VB phase output |
| 14 | Any feature that does not use TOUTm | S | | TOUT: W phase output |
| 15 | One-phase PWM output | S | Dead time (W phase) | TOUT: WB phase output |

Note 1. M: Master channel, S: Slave channel




Note 1. For the connection destination, see **Figure 29.25, Block Diagram: Motor Output Buffer Control**.

**Figure 29.43 Block Diagram: Delay Pulse Output with Dead Time**

## 29.13.3 Registers

### 29.13.3.1 PIC0REG2n2 — Timer I/O Control Register 2n2 (n = 0)

This register selects CHm input signals of the TAUDn timer. This section describes the bits to be used in the delay pulse output with dead time.

**Access:** This register can be read or written in 32-bit units.

**Address:** PIC0REG202: FFDD 00C8$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | PIC0REG2n227 | — | — | — | PIC0REG2n223 | — | — | — | PIC0REG2n219 | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R | R | R | R/W | R | R | R | R/W | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | PIC0REG2n204 | PIC0REG2n203 | PIC0REG2n202 | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R | R |

**Table 29.68 PIC0REG2n2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | Reserved | *1 |
| 27 | PIC0REG2n227 | Select the TIN input signal to TAUDTTIN15.<br>**PIC0REG2n227** / **Input signal**<br>1 / Signal selected by the PIC0REG2n204 bit<br>Other than above / Setting prohibited |
| 26 to 24 | Reserved | *1 |
| 23 | PIC0REG2n223 | Select the TIN input signal to TAUDTTIN13.<br>**PIC0REG2n223** / **Input signal**<br>1 / Signal selected by the PIC0REG2n203 bit<br>Other than above / Setting prohibited |
| 22 to 20 | Reserved | *1 |
| 19 | PIC0REG2n219 | Select the TIN input signal to TAUDTTIN11.<br>**PIC0REG2n219** / **Input signal**<br>1 / Signal selected by the PIC0REG2n202 bit<br>Other than above / Setting prohibited |
| 18 to 5 | Reserved | *1 |
| 4 | PIC0REG2n204 | Select the signal supplied to TAUDTTIN15.<br>0: Select TAUDTTOUT9.<br>1: Setting prohibited |
| 3 | PIC0REG2n203 | Select the signal supplied to TAUDTTIN13.<br>0: Select TAUDTTOUT7.<br>1: Setting prohibited |
| 2 | PIC0REG2n202 | Select the signal supplied to TAUDTTIN11.<br>0: Select TAUDTTOUT5.<br>1: Setting prohibited |
| 1, 0 | Reserved | *1 |

Note 1. Some of the bits defined as 0 in the PIC0REG2n2 register are defined for the other timer connection functions. For such bits, use the bit definition of those timer connection functions.

### 29.13.3.2 PIC0HIZCENn — Hi-Z Output Control Register n (n = 0)

This register selects the Hi-Z output control input signal of TAPAn.

**Access:** This register can be read or written in 8-bit units.

**Address:** PIC0HIZCEN0: FFDD $0080_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | PIC0HIZCENn6 | — | — | — | PIC0HIZCENn2 | — | PIC0HIZCENn0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R | R | R | R/W | R | R/W |

**Table 29.69 PIC0HIZCENn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 6 | PIC0HIZCENn6 | Select whether to enable or disable Hi-Z output control by the INTADCA0ERR interrupt signal.<br>0: Disable<br>1: Enable |
| 5 to 3 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 2 | PIC0HIZCENn2 | Select whether to enable or disable Hi-Z output control by the WDTA1NMI interrupt signal.<br>0: Disable<br>1: Enable |
| 1 | Reserved | When read, the value after reset is returned.<br>When writing, write the value after reset. |
| 0 | PIC0HIZCENn0 | Select whether to enable or disable Hi-Z output control by the TAPAnESO pin input.<br>0: Disable<br>1: Enable |

### 29.13.4 Operation Example

This is achieved by combining the following TAUD features:

- Delay pulse output function
- One-phase PWM output

The delay pulse output feature generates a PWM signal that is later than the cycle timing by an amount equal to the delay amount. Next, a one-phase PWM signal to which dead time has been added is output for the delayed PWM signal by one-phase PWM output feature.

A delay pulse with dead time is output by assigning the PWM output achieved using the above features to the U, V, and W phases. Therefore, the PWM output dead time can be freely specified for the PWM signal of each phase. Because the only difference between phases is the assigned channel, only one phase (the U phase) is described below.

#### 29.13.4.1 Delay pulse output function

By using a combination of CH2, CH4, and CH5, a basic PWM signal for one-phase PWM is output from TOUT05 delayed by the amount specified by CH4 with respect to the cycle specified by CH2.

Note that CH3 is a reserved timer for achieving this feature, so do not use it for other features.

**CAUTION**

**Do not specify a delay amount that exceeds the cycle.**

#### 29.13.4.2 One-phase PWM output

One-phase PWM output is generated from TOUT10 and TOUT11 by using a combination of CH10 and CH11.

By specifying the dead time value for CDR11, a one-phase PWM with dead time is output for the TIN11 input.

Similarly, the V phase uses CH12 and CH13 to output a one-phase PWM with dead time, while the W phase uses CH14 and CH15.

**CAUTION**

**Specify the same clock for each TAUDn channel that uses the delay pulse output and one-phase PWM output features.**

For details about the TAUD functions, see **Section 25, Timer Array Unit D (TAUD)**.

The differences between the delay pulse output with dead time and the three-phase PWM output with dead time are described below.

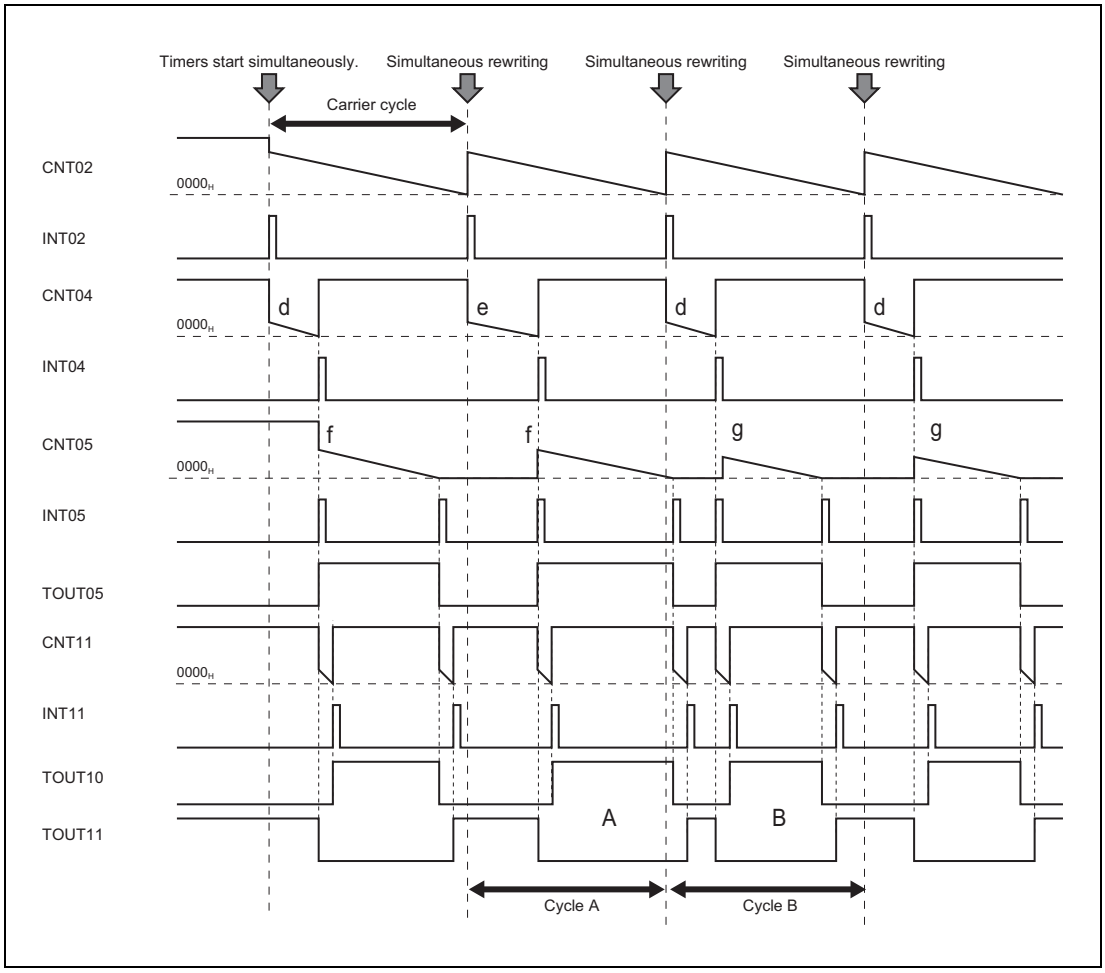



**Figure 29.44 PWM Output by Outputting a Delay Pulse with Dead Time**

In **Figure 29.44**, PWM waveform A is supposed to be output before cycle A ends, but because the delay timing is too long, the PWM clear position is after the end of cycle A. Next, PWM waveform B, which is for cycle B, is output.

The operations shown below occur when an attempt is made to achieve the operations shown in **Figure 29.44, PWM Output by Outputting a Delay Pulse with Dead Time** by the three-phase PWM output with dead time.

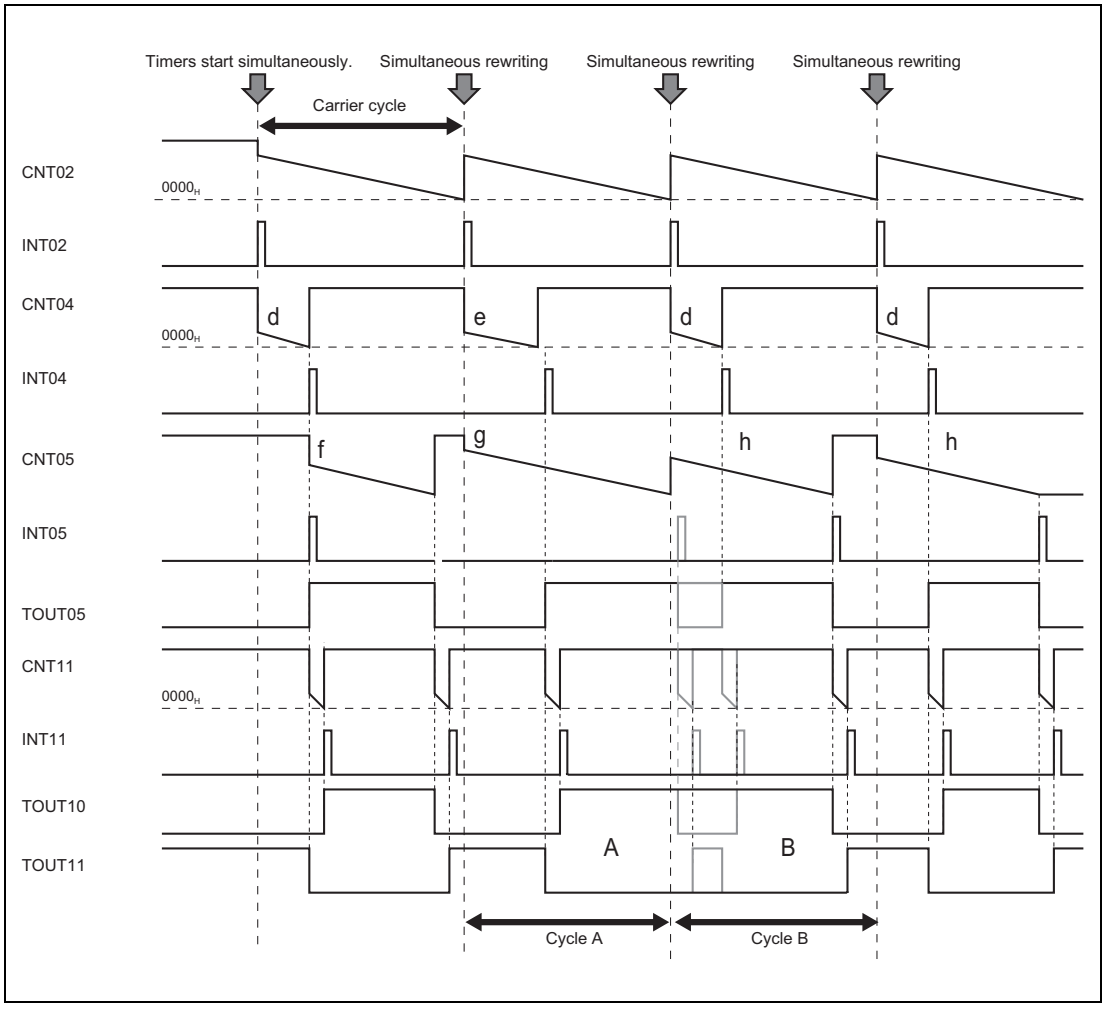



**Figure 29.45 Output of a Three-Phase PWM Signal with Dead Time (1)**

**Figure 29.45** shows an example in which the output PWM signal does not end before carrier cycle A because the set timing for outputting a three-phase PWM signal with dead time is delayed and the clear timing is after the end of the carrier cycle.

For cycle A, the set timing of PWM waveform A is the same as that in the figure on the previous page, but because the clear timing is after the end of cycle A, a reload operation occurs in cycle A before PWM waveform A is cleared, and the clear timing for PWM waveform A does not occur.

In addition, the set timing of PWM waveform B for cycle B is ignored because a PWM waveform is already set. The result is that there is no PWM waveform change until the clear timing of cycle B, and a waveform that combines PWM waveform A and PWM waveform B is output.

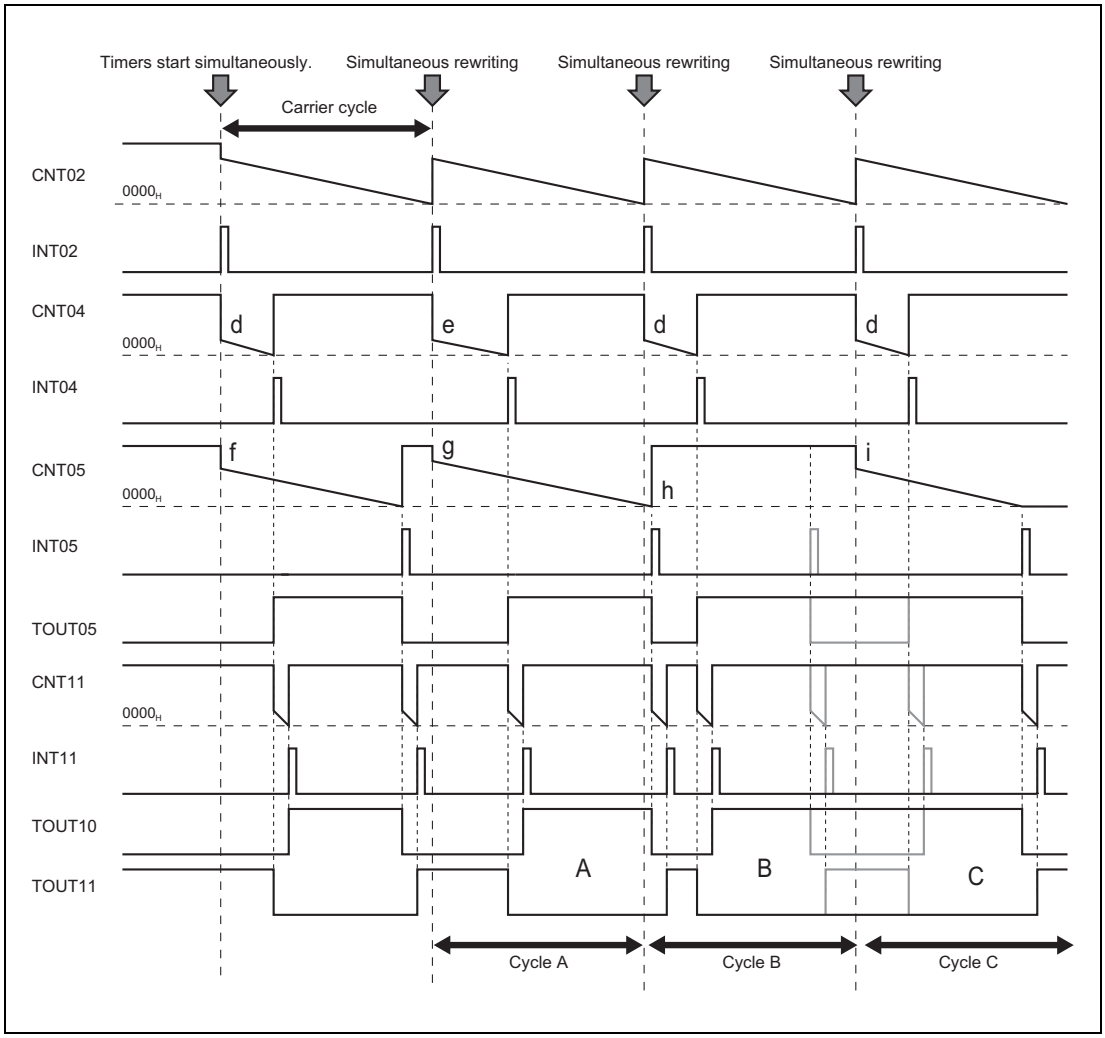



**Figure 29.46 Output of a Three-Phase PWM Signal with Dead Time (2)**

**Figure 29.46** shows an example of outputting a three-phase PWM signal with dead time in which counter operation for which the clear timing is longer than cycle A is continued in cycle B, and PWM output A is cleared at the beginning of cycle B.

The output of PWM waveform A for cycle A is the same as the output of a delay pulse with dead time, but because the clear timing is used at the beginning of cycle B, the clear timing of PWM output B, which is supposed to be output during cycle B, does not occur.

In addition, the set timing of PWM waveform C for cycle C is ignored because a PWM waveform is already set. The result is that there is no PWM waveform change until the clear timing of cycle C, and a waveform that combines PWM waveform B and PWM waveform C is output.

In this way, it is possible to achieve freer PWM output timing when outputting a delay pulse with dead time than when outputting a three-phase PWM signal with dead time.

The peripheral interconnections provide a connection for using the PWM output timing of delay pulse output as input for one-phase PWM output.

**Figure 29.47** shows a timing chart for outputting a delay pulse with dead time.

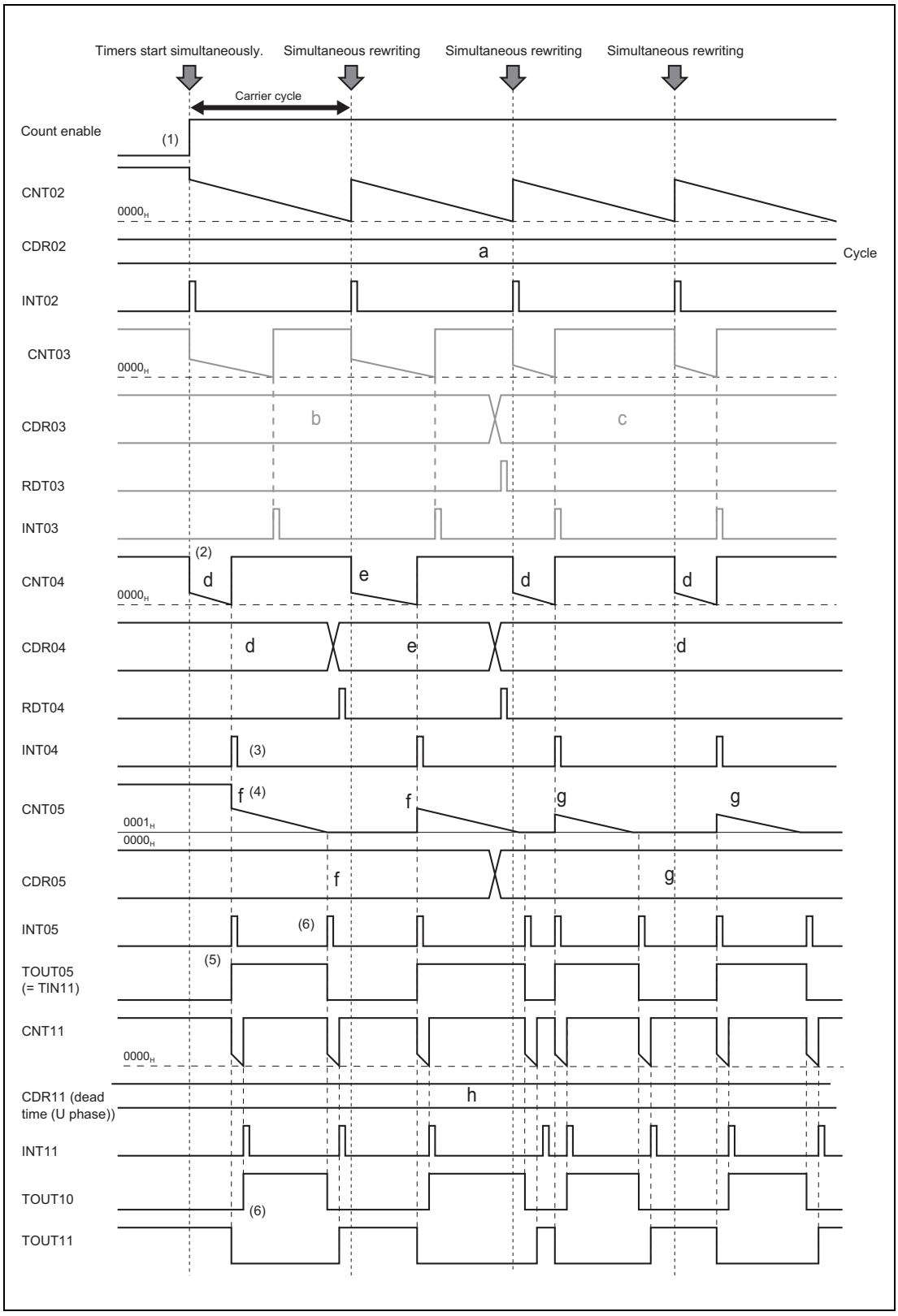



**Figure 29.47 Output of a Delay Pulse with Dead Time**

The output of a delay pulse with dead time shown in **Figure 29.47** is described below.

(1) CH2 (the carrier cycle timer) and CH4 (the delay timing timer) are started simultaneously by starting timers simultaneously.
CH5 (the PWM duty timer) and CH11 (the dead time timer) are also enabled, but no counting operations are performed until the edges of INT04, which indicates the count start timing for CH5, and TIN11, which indicates the count start timing for CH11, are detected.
Because CH3 does not affect PWM output for this function, the channel is not described.

(2) For CH4, when there is a CH2 underflow, the settings from CDR04 are reloaded to CNT04.

(3) The CH4 underflow generates the delay timing signal (INT04).

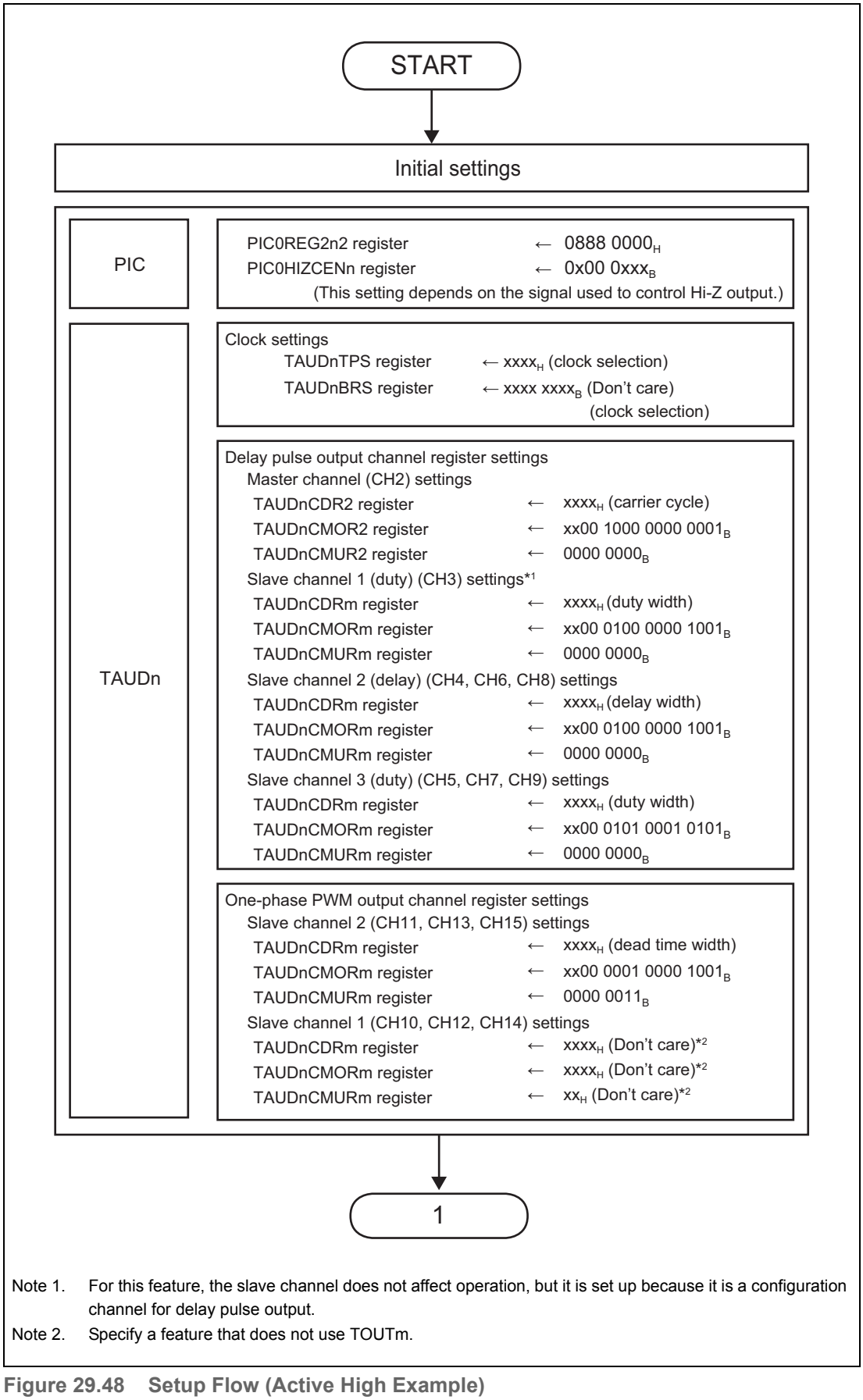
(4) When INT04 is generated, the settings from CDR05 are reloaded to CNT05, and then the CH5 (the PWM duty timer) operation starts.

(5) At this time, INT05 is generated and the TOUT05 output level changes to the active level.

(6) Due to the CH5 underflow, INT05 is generated again, and TOUT05 changes to the inactive level. TOUT05, which is changed by the CH4 and CH5 underflow, is supplied to the TIN11 input of one-phase PWM output.

(7) During one-phase PWM output, a PWM waveform with dead time is output by detecting a TIN11 edge.

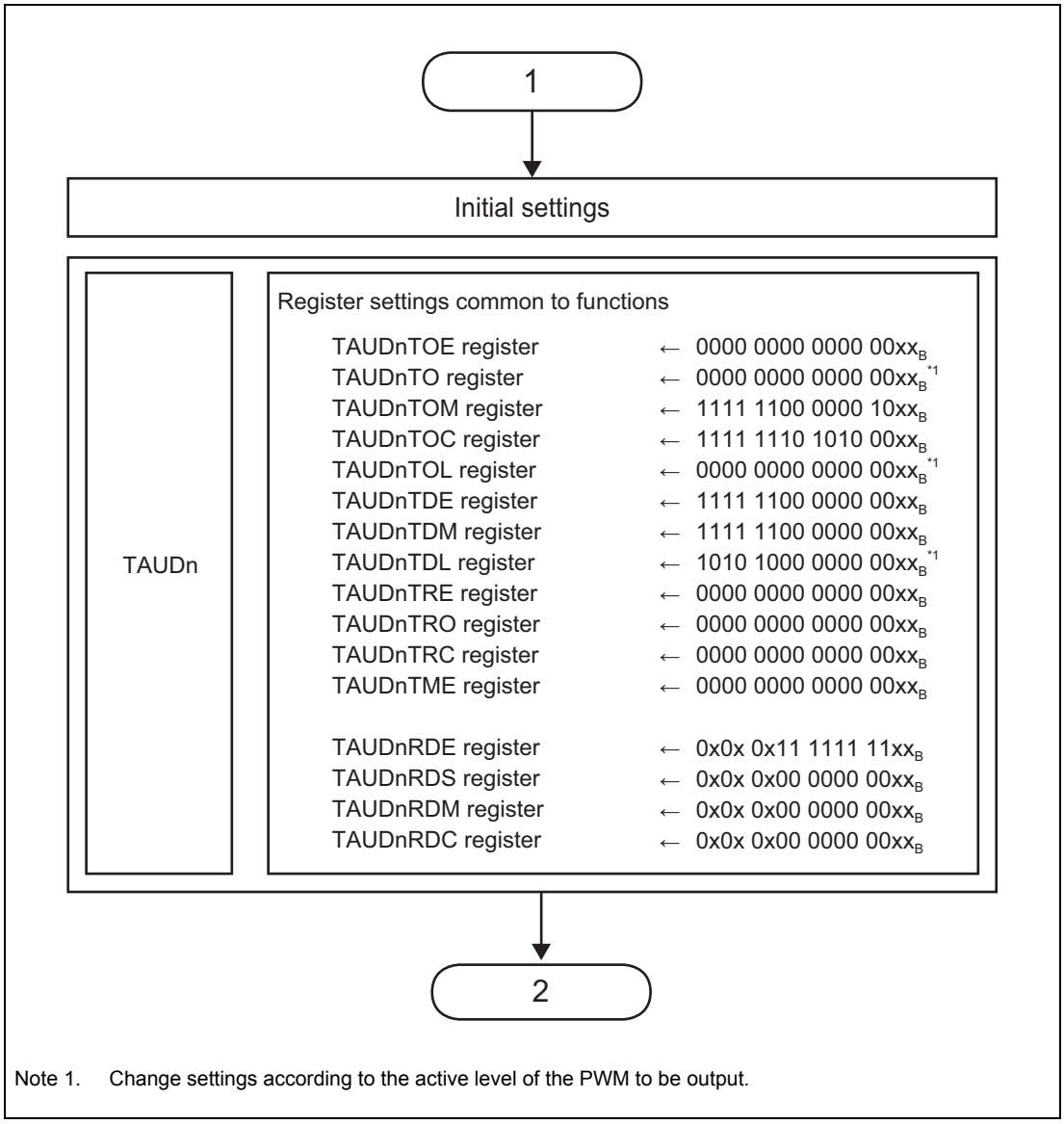
### 29.13.5 Setup Flow




Note 1. For this feature, the slave channel does not affect operation, but it is set up because it is a configuration channel for delay pulse output.

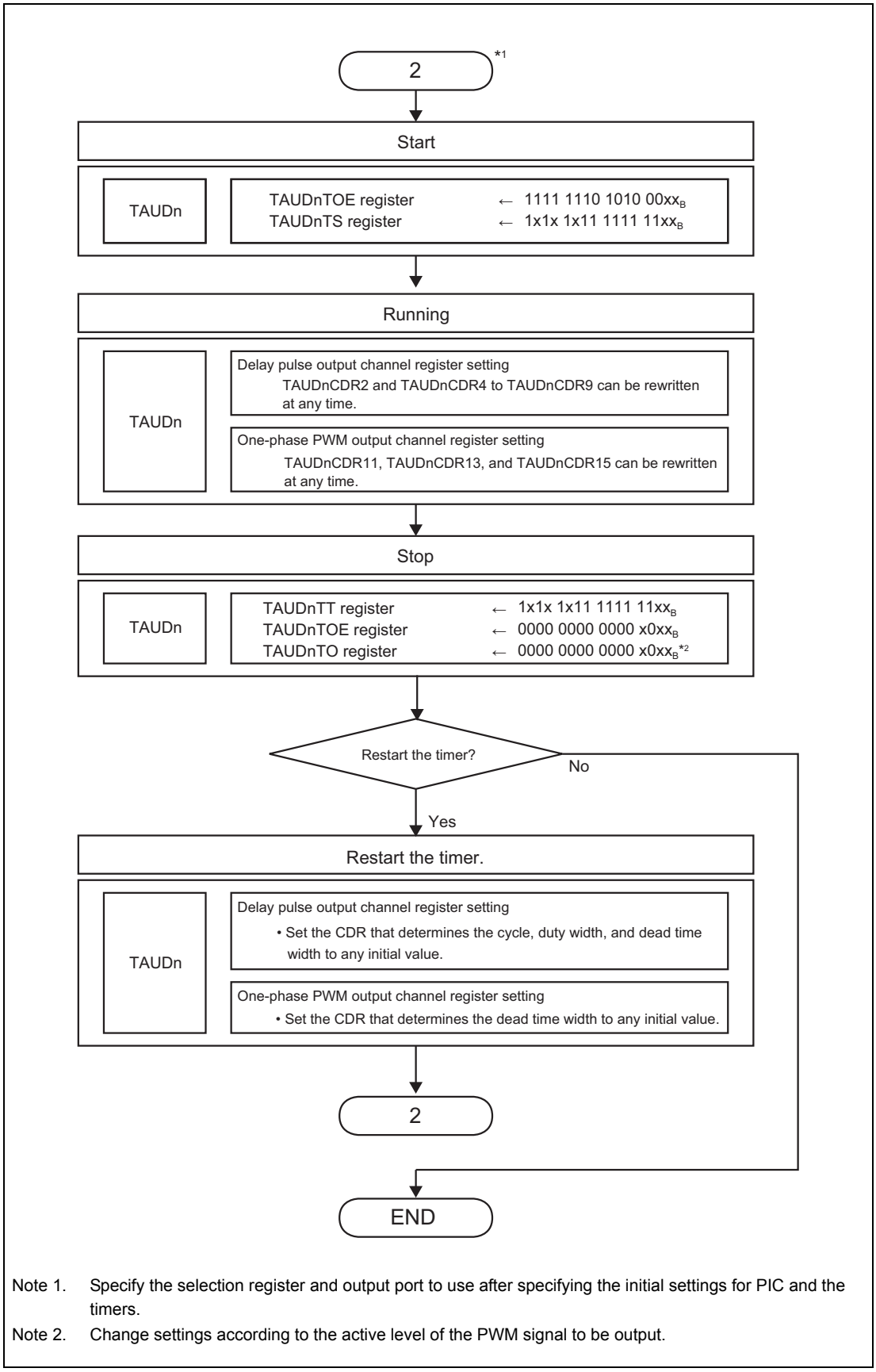
Note 2. Specify a feature that does not use TOUTm.

**Figure 29.48 Setup Flow (Active High Example)**




Note 1. Change settings according to the active level of the PWM to be output.

**Figure 29.49 Setup Flow (Active High Example) (continued)**




Note 1. Specify the selection register and output port to use after specifying the initial settings for PIC and the timers.

Note 2. Change settings according to the active level of the PWM signal to be output.

**Figure 29.50 Setup Flow (Active High Example) (continued)**

## 29.13.6 Setting Examples for Operation Functions

This section provides example settings for each register.

### 29.13.6.1 TAUDn Settings

**Table 29.70 TAUDn: CH2-related (Master Channel used to Output a Delay Pulse*1)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnCMOR2 | 15, 14 | TAUDnCKS[1:0] | Don't care*2 | Operation clock setting |
| | 13, 12 | TAUDnCCS[1:0] | 00 | |
| | 11 | TAUDnMAS | 1 | |
| | 10 to 8 | TAUDnSTS[2:0] | 000 | |
| | 7, 6 | TAUDnCOS[1:0] | 00 | |
| | 5 | | 0 | Fixed to 0 |
| | 4 to 1 | TAUDnMD[4:1] | 0000 | |
| | 0 | TAUDnMD0 | 1 | Output INTm at the start of operation. |
| TAUDnCMUR2 | 1, 0 | TAUDnTIS[1:0] | 00 | |

Note 1. The master channel and slave channel names are defined for TAUD delay pulse output. For details, see **Section 25, Timer Array Unit D (TAUD)**.

Note 2. The same operation clock must be specified for the master channel and slave channel.

**Table 29.71 TAUDn: CH3-related (Slave Channel used to Output a Delay Pulse*1*2)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnCMOR3 | 15, 14 | TAUDnCKS[1:0] | Don't care*3 | Operation clock setting |
| | 13, 12 | TAUDnCCS[1:0] | 00 | |
| | 11 | TAUDnMAS | 0 | |
| | 10 to 8 | TAUDnSTS[2:0] | 100 | Start trigger: INTm detection on the master channel |
| | 7, 6 | TAUDnCOS[1:0] | 00 | |
| | 5 | | 0 | Fixed to 0 |
| | 4 to 1 | TAUDnMD[4:1] | 0100 | |
| | 0 | TAUDnMD0 | 1 | Enable start triggers during counting. |
| TAUDnCMUR3 | 1, 0 | TAUDnTIS[1:0] | 00 | |

Note 1. The master channel and slave channel names are defined for TAUD delay pulse output. For details, see **Section 25, Timer Array Unit D (TAUD)**.

Note 2. The same operation clock must be specified for the master channel and slave channel.

Note 3. For this feature, the channel does not affect operation, but it is set up because it is a configuration channel for delay pulse output.

**NOTE**

For the TAUDnCMORm register used during delay pulse output, TAUDnCKS[1:0] (which selects the operation clock) can be set to any value, but other control bits have fixed values. For details, see **Section 25, Timer Array Unit D (TAUD)**.

**Table 29.72 TAUDn: CH4, CH6, and CH8-related (Slave Channel 2 used to Output a Delay Pulse*1) (m = 4, 6, or 8)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnCMORm | 15, 14 | TAUDnCKS[1:0] | Don't care*2 | Operation clock setting |
| | 13, 12 | TAUDnCCS[1:0] | 00 | |
| | 11 | TAUDnMAS | 0 | |
| | 10 to 8 | TAUDnSTS[2:0] | 100 | Start trigger: INTm detection on the master channel |
| | 7, 6 | TAUDnCOS[1:0] | 00 | |
| | 5 | | 0 | Fixed to 0 |
| | 4 to 1 | TAUDnMD[4:1] | 0100 | |
| | 0 | TAUDnMD0 | 1 | Enable start triggers during counting. |
| TAUDnCMURm | 1, 0 | TAUDnTIS[1:0] | 00 | |

Note 1. The master channel and slave channel names are defined for TAUD delay pulse output. For details, see **Section 25, Timer Array Unit D (TAUD)**.

Note 2. The same operation clock must be specified for the slave channel and master channel.

**NOTE**

For the TAUDnCMORm register used during delay pulse output, TAUDnCKS[1:0] (which selects the operation clock) can be set to any value, but other control bits have fixed values. For details, see **Section 25, Timer Array Unit D (TAUD)**.

**Table 29.73 TAUDn: CH5, CH7, and CH9-related (Slave Channel 3 used to Output a Delay Pulse*1) (m = 5, 7, or 9)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnCMORm | 15, 14 | TAUDnCKS[1:0] | Don't care*2 | Operation clock setting |
| | 13, 12 | TAUDnCCS[1:0] | 00 | |
| | 11 | TAUDnMAS | 0 | |
| | 10 to 8 | TAUDnSTS[2:0] | 101 | Start trigger: INTm detection on an upper channel |
| | 7, 6 | TAUDnCOS[1:0] | 00 | |
| | 5 | | 0 | Fixed to 0 |
| | 4 to 1 | TAUDnMD[4:1] | 1010 | |
| | 0 | TAUDnMD0 | 1 | Enable start triggers during counting. |
| TAUDnCMURm | 1, 0 | TAUDnTIS[1:0] | 00 | |

Note 1. The master channel and slave channel names are defined for TAUD delay pulse output. For details, see **Section 25, Timer Array Unit D (TAUD)**.

Note 2. The same operation clock must be specified for the slave channel and master channel.

**NOTE**

For the TAUDnCMORm register used during delay pulse output, TAUDnCKS[1:0] (which selects the operation clock) can be set to any value, but other control bits have fixed values. For details, see **Section 25, Timer Array Unit D (TAUD)**.

**Table 29.74 TAUDn: CH11, CH13, and CH15-related (One-Phase PWM Output) (m = 11, 13, or 15)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnCMORm | 15, 14 | TAUDnCKS[1:0] | Don't care*1 | Operation clock setting |
| | 13, 12 | TAUDnCCS[1:0] | 00 | |
| | 11 | TAUDnMAS | 0 | |
| | 10 to 8 | TAUDnSTS[2:0] | 001 | Start trigger: Detection of a TINm-input valid edge |
| | 7, 6 | TAUDnCOS[1:0] | 00 | |
| | 5 | | 0 | Fixed to 0 |
| | 4 to 1 | TAUDnMD[4:1] | 0100 | |
| | 0 | TAUDnMD0 | 1 | Enable start triggers during counting. |
| TAUDnCMURm | 1, 0 | TAUDnTIS[1:0] | 11 | Both rising and falling TINm edges are detected as valid. (High width) |

Note 1. Specify the same operation clock settings as for the PWM output master channel (CH2).

**NOTE**

For the TAUDnCMORm register used during one-phase PWM output, TAUDnCKS[1:0] (which selects the operation clock) can be set to any value, but other control bits have fixed values. For details, see **Section 25, Timer Array Unit D (TAUD)**.
CH10, CH12, and CH14 can be used with any feature that does not use TOUTm output (such as A/D trigger output).

**Table 29.75 Common TAUDn Channel Settings (1/4)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnTOE | 15 to 10 | TAUDnTOE15 to TAUDnTOE10 | 0<br>1 | Disable the timer.<br>Enable the timer. |
| | 9 | TAUDnTOE09 | 0<br>1 | Disable the timer.<br>Enable the timer. |
| | 8 | TAUDnTOE08 | 0 | This is fixed to 0 because TOUT08 is not used. |
| | 7 | TAUDnTOE07 | 0<br>1 | Disable the timer.<br>Enable the timer. |
| | 6 | TAUDnTOE06 | 0 | This is fixed to 0 because TOUT06 is not used |
| | 5 | TAUDnTOE05 | 0<br>1 | Disable the timer.<br>Enable the timer. |
| | 4 | TAUDnTOE04 | 0 | This is fixed to 0 because TOUT04 is not used |
| | 3 | TAUDnTOE03 | 0 | This is fixed to 0 because TOUT03 is not used |
| | 2 | TAUDnTOE02 | 0 | This is fixed to 0 because TOUT02 is not used. |
| | 1, 0 | TAUDnTOE01<br>TAUDnTOE00 | Don't care | |
| TAUDnTO | 15 to 10 | TAUDnTO15 to TAUDnTO10 | 0*1 | Output a low-level signal to TOUT15 to TOUT10. |
| | 9 to 2 | TAUDnTO09 to TAUDnTO02 | 0 | Output a low-level signal to TOUT09 to TOUT02. |
| | 1, 0 | TAUDnTO01<br>TAUDnTO00 | Don't care | |

**Table 29.75 Common TAUDn Channel Settings (2/4)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnTOM | 15 to 10 | TAUDnTOM15 to TAUDnTOM10 | 1 | Synchronous operation mode |
| | 9 to 4 | TAUDnTOM09 to TAUDnTOM04 | 0 | Independent operation mode |
| | 3 | TAUDnTOM03 | 1 | Synchronous operation mode |
| | 2 | TAUDnTOM02 | 0 | Independent operation mode |
| | 1, 0 | TAUDnTOM01 TAUDnTOM00 | Don't care | |
| TAUDnTOC | 15 to 10 | TAUDnTOC15 to TAUDnTOC10 | 1 | Synchronous operation mode 2 |
| | 9 to 4 | TAUDnTOC09 to TAUDnTOC04 | 1, 0, 1, 0, 1, 0 | CH5, CH7, CH9: Operation mode 2<br>CH4, CH6, CH8: Operation mode 1 |
| | 3 | TAUDnTOC03 | 0 | Operation mode 1 |
| | 2 | TAUDnTOC02 | 0 | Operation mode 1 |
| | 1, 0 | TAUDnTOC01 TAUDnTOC00 | Don't care | |
| TAUDnTOL | 15 to 10 | TAUDnTOL15 to TAUDnTOL10 | 0*1 | Positive logic output (active high) |
| | 9 to 2 | TAUDnTOL09 to TAUDnTOL02 | 0 | Positive logic output (active high) |
| | 1, 0 | TAUDnTOL01 TAUDnTOL00 | Don't care | |
| TAUDnTDE | 15 to 10 | TAUDnTDE15 to TAUDnTDE10 | 1 | Enable dead time control.*2 |
| | 9 to 2 | TAUDnTDE09 to TAUDnTDE02 | 0 | Disable dead time control. |
| | 1, 0 | TAUDnTDE01 TAUDnTDE00 | Don't care | |
| TAUDnTDM | 15 to 10 | TAUDnTDM15 to TAUDnTDM10 | 1 | Output dead time upon detecting a TINm input edge at a lower odd channel. |
| | 9 to 2 | TAUDnTDM09 to TAUDnTDM02 | 0 | Invalid because dead time control is disabled. |
| | 1, 0 | TAUDnTDM01 TAUDnTDM00 | Don't care | |
| TAUDnTDL | 15 | TAUDnTDL15 | 1*1 | Add dead time to the negative W phase period. |
| | 14 | TAUDnTDL14 | 0*1 | Add dead time to the positive W phase period. |
| | 13 | TAUDnTDL13 | 1*1 | Add dead time to the negative V phase period. |
| | 12 | TAUDnTDL12 | 0*1 | Add dead time to the positive V phase period. |
| | 11 | TAUDnTDL11 | 1*1 | Add dead time to the negative U phase period. |
| | 10 | TAUDnTDL10 | 0*1 | Add dead time to the positive U phase period. |
| | 9 to 2 | TAUDnTDL09 to TAUDnTDL02 | 0 | Invalid because dead time control is disabled. |
| | 1, 0 | TAUDnTDL01 TAUDnTDL00 | Don't care | |
| TAUDnTRE | 15 to 2 | TAUDnTRE15 to TAUDnTRE02 | 0 | Disable real-time output. |
| | 1, 0 | TAUDnTRE01 TAUDnTRE00 | Don't care | |

**Table 29.75 Common TAUDn Channel Settings (3/4)**

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnTRO | 15 to 2 | TAUDnTRO15 to TAUDnTRO02 | 0 | Invalid because real-time output is disabled. |
| | 1, 0 | TAUDnTRO01 TAUDnTRO00 | Don't care | |
| TAUDnTRC | 15 to 2 | TAUDnTRC15 to TAUDnTRC02 | 0 | Do not use this channel to generate the real-time output trigger. |
| | 1, 0 | TAUDnTRC01 TAUDnTRC00 | Don't care | |
| TAUDnTME | 15 to 2 | TAUDnTME15 to TAUDnTME02 | 0 | Disable modulation output for timer output and real-time output. |
| | 1, 0 | TAUDnTME01 TAUDnTME00 | Don't care | |
| TAUDnRDE | 15 | TAUDnRDE15 | 0 | Disable simultaneous rewriting. |
| | 14 | TAUDnRDE14 | Don't care | |
| | 13 | TAUDnRDE13 | 0 | Disable simultaneous rewriting. |
| | 12 | TAUDnRDE12 | Don't care | |
| | 11 | TAUDnRDE11 | 0 | Disable simultaneous rewriting. |
| | 10 | TAUDnRDE10 | Don't care | |
| | 9 to 2 | TAUDnRDE09 to TAUDnRDE02 | 1 | Enable simultaneous rewriting. |
| | 1, 0 | TAUDnRDE01 TAUDnRDE00 | Don't care | |
| TAUDnRDS | 15 | TAUDnRDS15 | 0 | Do not enable simultaneous rewriting by using another upper channel. |
| | 14 | TAUDnRDS14 | Don't care | |
| | 13 | TAUDnRDS13 | 0 | Do not enable simultaneous rewriting by using another upper channel. |
| | 12 | TAUDnRDS12 | Don't care | |
| | 11 | TAUDnRDS11 | 0 | Do not enable simultaneous rewriting by using another upper channel. |
| | 10 | TAUDnRDS10 | Don't care | |
| | 9 to 2 | TAUDnRDS09 to TAUDnRDS02 | 0 | Enable simultaneous rewriting by using a master channel. |
| | 1, 0 | TAUDnRDS01 TAUDnRDS00 | Don't care | |
| TAUDnRDM | 15 | TAUDnRDM15 | 0 | Invalid because simultaneous rewriting is not enabled. |
| | 14 | TAUDnRDM14 | Don't care | |
| | 13 | TAUDnRDM13 | 0 | Invalid because simultaneous rewriting is not enabled. |
| | 12 | TAUDnRDM12 | Don't care | |
| | 11 | TAUDnRDM11 | 0 | Invalid because simultaneous rewriting is not enabled. |
| | 10 | TAUDnRDM10 | Don't care | |
| | 9 to 2 | TAUDnRDM09 to TAUDnRDM02 | 0 | Load the signal when the master channel starts counting. |
| | 1, 0 | TAUDnRDM01 TAUDnRDM00 | Don't care | |

Table 29.75 Common TAUDn Channel Settings (4/4)

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| TAUDnRDC | 15 | TAUDnRDC15 | 0 | Invalid because simultaneous rewriting is not enabled. |
| | 14 | TAUDnRDC14 | Don't care | |
| | 13 | TAUDnRDC13 | 0 | Invalid because simultaneous rewriting is not enabled. |
| | 12 | TAUDnRDC12 | Don't care | |
| | 11 | TAUDnRDC11 | 0 | Invalid because simultaneous rewriting is not enabled. |
| | 10 | TAUDnRDC10 | Don't care | |
| | 9 to 2 | TAUDnRDC09 to TAUDnRDC02 | 0 | Do not use this channel to generate the simultaneous rewrite trigger. |
| | 1, 0 | TAUDnRDC01 TAUDnRDC00 | Don't care | |

Note 1. Change the setting according to the used system.

Note 2. These are used to control positive/negative phase waveform output for which even channels are paired with odd channels to perform dead time control. For details, see **Section 25, Timer Array Unit D (TAUD)**.

### 29.13.6.2 Peripheral Interconnections Settings

Table 29.76 Peripheral Interconnections Settings

| Register | Bit position | Bit name | Setting | Remark |
|---|---|---|---|---|
| PIC0REG2n2 | 27 | PIC0REG2n227 | 1 | Select the input selected by the PIC0REG2n204 bit. |
| | 23 | PIC0REG2n223 | 1 | Select the input selected by the PIC0REG2n203 bit. |
| | 19 | PIC0REG2n219 | 1 | Select the input selected by the PIC0REG2n202 bit. |
| | 4 | PIC0REG2n204 | 0 | Select TAUDTTOUT9. |
| | 3 | PIC0REG2n203 | 0 | Select TAUDTTOUT7. |
| | 2 | PIC0REG2n202 | 0 | Select TAUDTTOUT5. |

# Section 30 PWM Output/Diagnostic (PWM-Diag)

This section contains a generic description of the PWM output/diagnostic function (PWM-Diag).

The first part of this section describes all RH850/F1K specific properties, such as the number of units, register base addresses, etc. The remainder of the section describes the functions and registers of the units constituting PWM-Diag.

## 30.1 Features of RH850/F1K PWM-Diag

### 30.1.1 Number of Units and Channels

The PWM-Diag unit consists of a PWBA block for generating clock signals, PWGA blocks that generate PWM signals, and a PWSA block for generating triggers for A/D conversion. The numbers of individual units are listed below.

Each PWGA unit has one PWM channel interface. “Number of channels” is used with the same meaning as “number of units” in this section.

**Table 30.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| **PWBA** | | | |
| Number of Units | 1 | | |
| Name | PWBAn (n = 0) | | |
| **PWGA** | | | |
| Number of Units | 48 | 64 | 72 |
| Name | PWGAn (n = 0 to 47) | PWGAn (n = 0 to 63) | PWGAn (n = 0 to 71) |
| **PWSA** | | | |
| Number of Units | 1 | | |
| Name | PWSAn (n = 0) | | |

**Table 30.2 Indices**

| Index | Description |
|---|---|
| n | Throughout this section, individual units constituting the PWM-Diag function are identified by the index “n”; for example, PWBAnTE indicates the PWBAn status register. |
| m | The PWBA generation clock is identified by the index “m” (m = 0 to 3); for example, PWBAnBRSm indicates the PWMCLKm clock cycle configuration register. |
| x, y | An A/D converter configuration register number corresponding to a PWM-Diag channel is identified by the index “x, y”; for example, PWSAnPVCRx_y (x_y = 00_01, 02_03, ..., 70_71). |
| j | Registers storing trigger channel numbers (encoded value) from PWGAn are identified by the index “j”; for example, the PWSAnQUEj register (j = 0 to 7). |
| k | Sets of registers where each has the same function are identified by the index “k”; for example, the SLPWGAk register (k = 0 to 2). |

The following table shows values indicated by the indices of each product.

**Table 30.3 Indices of Products**

| Indices of each product | | |
|---|---|---|
| **100 pins** | **144 pins** | **176 pins** |
| x = 00, 02, ,,, 46<br>y = 01, 03, ,,, 47 | x = 00, 02, ,,, 62,<br>y = 01, 03, ,,, 63 | x = 00, 02, ,,, 70<br>y = 01, 03, ,,, 71 |
| j = 0 to 7 | j = 0 to 7 | j = 0 to 7 |
| k = 0, 1 | k = 0, 1 | k = 0 to 2 |

## 30.1.2 Register Base Address

PWM-Diag base addresses are listed in the following table.

PWM-Diag register addresses are given as offsets from the base addresses.

**Table 30.4 Register Base Addresses**

| Base Address Name | Base Address |
|---|---|
| <PWBAn_base> | FFE7 $2800_H$ |
| <PWGAn_base> | FFE7 $1000_H$ + $40_H$ × n |
| <PWSAn_base> | FFE7 $0000_H$ |
| <SLPW_base> | FFE7 $3000_H$ |

## 30.1.3 Clock Supply

The PWM-Diag clock supply is shown in the following table.

**Table 30.5 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| PWBAn | PCLK | CKSCLK_IPERI2 | Wave form clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CKSCLK_IPERI2 | |
| PWGAn | PCLK | CKSCLK_IPERI2 | Wave form clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CKSCLK_IPERI2 | |
| PWSAn | PCLK | CKSCLK_IPERI2 | Wave form clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CKSCLK_IPERI2 | |

### 30.1.4 Interrupt Requests

PWM-Diag interrupt requests are listed in the following table.

Table 30.6 Interrupt Requests (1/2)

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| PWGA_INT0 | PWGA0 interrupt | 92 | — |
| PWGA_INT1 | PWGA1 interrupt | 93 | — |
| PWGA_INT2 | PWGA2 interrupt | 94 | — |
| PWGA_INT3 | PWGA3 interrupt | 95 | — |
| PWGA_INT4 | PWGA4 interrupt | 85 | — |
| PWGA_INT5 | PWGA5 interrupt | 86 | — |
| PWGA_INT6 | PWGA6 interrupt | 87 | — |
| PWGA_INT7 | PWGA7 interrupt | 88 | — |
| PWGA_INT8 | PWGA8 interrupt | 96 | — |
| PWGA_INT9 | PWGA9 interrupt | 97 | — |
| PWGA_INT10 | PWGA10 interrupt | 98 | — |
| PWGA_INT11 | PWGA11 interrupt | 99 | — |
| PWGA_INT12 | PWGA12 interrupt | 100 | — |
| PWGA_INT13 | PWGA13 interrupt | 101 | — |
| PWGA_INT14 | PWGA14 interrupt | 102 | — |
| PWGA_INT15 | PWGA15 interrupt | 103 | — |
| PWGA_INT16 | PWGA16 interrupt | 145 | — |
| PWGA_INT17 | PWGA17 interrupt | 147 | — |
| PWGA_INT18 | PWGA18 interrupt | 149 | — |
| PWGA_INT19 | PWGA19 interrupt | 151 | — |
| PWGA_INT20 | PWGA20 interrupt | 124 | — |
| PWGA_INT21 | PWGA21 interrupt | 125 | — |
| PWGA_INT22 | PWGA22 interrupt | 126 | — |
| PWGA_INT23 | PWGA23 interrupt | 127 | — |
| PWGA_INT24 | PWGA24 interrupt | 184 | — |
| PWGA_INT25 | PWGA25 interrupt | 185 | — |
| PWGA_INT26 | PWGA26 interrupt | 153 | — |
| PWGA_INT27 | PWGA27 interrupt | 186 | — |
| PWGA_INT28 | PWGA28 interrupt | 187 | — |
| PWGA_INT29 | PWGA29 interrupt | 188 | — |
| PWGA_INT30 | PWGA30 interrupt | 155 | — |
| PWGA_INT31 | PWGA31 interrupt | 157 | — |
| PWGA_INT32 | PWGA32 interrupt | 189 | — |
| PWGA_INT33 | PWGA33 interrupt | 190 | — |
| PWGA_INT34 | PWGA34 interrupt | 191 | — |
| PWGA_INT35 | PWGA35 interrupt | 192 | — |
| PWGA_INT36 | PWGA36 interrupt | 193 | — |
| PWGA_INT37 | PWGA37 interrupt | 194 | — |
| PWGA_INT38 | PWGA38 interrupt | 195 | — |
| PWGA_INT39 | PWGA39 interrupt | 196 | — |

Table 30.6 Interrupt Requests (2/2)

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| PWGA_INT40 | PWGA40 interrupt | 197 | — |
| PWGA_INT41 | PWGA41 interrupt | 198 | — |
| PWGA_INT42 | PWGA42 interrupt | 199 | — |
| PWGA_INT43 | PWGA43 interrupt | 200 | — |
| PWGA_INT44 | PWGA44 interrupt | 201 | — |
| PWGA_INT45 | PWGA45 interrupt | 202 | — |
| PWGA_INT46 | PWGA46 interrupt | 203 | — |
| PWGA_INT47 | PWGA47 interrupt | 204 | — |
| PWGA_INT48 | PWGA48 interrupt | 240 | — |
| PWGA_INT49 | PWGA49 interrupt | 241 | — |
| PWGA_INT50 | PWGA50 interrupt | 242 | — |
| PWGA_INT51 | PWGA51 interrupt | 243 | — |
| PWGA_INT52 | PWGA52 interrupt | 244 | — |
| PWGA_INT53 | PWGA53 interrupt | 245 | — |
| PWGA_INT54 | PWGA54 interrupt | 246 | — |
| PWGA_INT55 | PWGA55 interrupt | 247 | — |
| PWGA_INT56 | PWGA56 interrupt | 248 | — |
| PWGA_INT57 | PWGA57 interrupt | 249 | — |
| PWGA_INT58 | PWGA58 interrupt | 250 | — |
| PWGA_INT59 | PWGA59 interrupt | 251 | — |
| PWGA_INT60 | PWGA60 interrupt | 252 | — |
| PWGA_INT61 | PWGA61 interrupt | 253 | — |
| PWGA_INT62 | PWGA62 interrupt | 254 | — |
| PWGA_INT63 | PWGA63 interrupt | 255 | — |
| PWGA_INT64 | PWGA64 interrupt | 277 | — |
| PWGA_INT65 | PWGA65 interrupt | 278 | — |
| PWGA_INT66 | PWGA66 interrupt | 279 | — |
| PWGA_INT67 | PWGA67 interrupt | 280 | — |
| PWGA_INT68 | PWGA68 interrupt | 281 | — |
| PWGA_INT69 | PWGA69 interrupt | 282 | — |
| PWGA_INT70 | PWGA70 interrupt | 283 | — |
| PWGA_INT71 | PWGA71 interrupt | 284 | — |
| PWSA_INT_QFULL | PWSA queue full interrupt | 91 | — |

### 30.1.5 Reset Sources

PWM-Diag reset sources are listed in the following table. The individual PWM-Diag units are initialized by these reset sources.

**Table 30.7 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| PWBAn<br>PWGAn<br>PWSAn | All reset sources (ISORES) |

### 30.1.6 External Input/Output Signals

External input/output signals of the PWM-Diag are listed below.

**Table 30.8 External Input/Output Signals**

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| PWGA_TOUTn<br>(Unit: PWGA) | PWGA unit n output | PWGAnO |

**CAUTION**

When port P8_6 is used as PWGA38O, port P8_6 pin outputs a low-level $\overline{\text{RESETOUT}}$ signal while a reset is asserted and continues to output a low level after the reset is deasserted.
For details, see **Section 2.11.1.1, P8_6: $\overline{\text{RESETOUT}}$**.

### 30.1.7 Internal Signals

The I/O signals for connecting two PWM-Diag channels or a PWM-Diag and another function are listed below.

**Table 30.9 Internal Output Signals**

| Unit Signal Name | Description | Connected to |
|---|---|---|
| **PWBA0** | | |
| PWMCLK0 | PWGA count clock 0 | PWGAn |
| PWMCLK1 | PWGA count clock 1 | PWGAn |
| PWMCLK2 | PWGA count clock 2 | PWGAn |
| PWMCLK3 | PWGA count clock 3 | PWGAn |
| **PWGAn** | | |
| PWGA_TRGOUTn | PWGAn trigger | PWSA0 |
| **PWSA0** | | |
| PWSA_ADTRG[1:0] | A/D converter unit select signal | ADCA0, ADCA1 |
| PWSA_PVCR_VALUE[11:0] | A/D converter control signal | ADCA0, ADCA1 |
| **ADCAn** | | |
| ADC_CONV_ENDn | A/D conversion completion signal | PWSA0 |

### 30.1.8 Functional Overview

This function is comprised of four types of units: clock divider (PWBA), PWM generator (PWGA), A/D conversion trigger select function (PWSA), and A/D converter (ADCA).

#### PWBA

- Clock divider
  PWBA generates a PWMCLKm count clock signal by frequency division of PCLK and supplies it to the PWM generator PWGA.
  The cycle of the PWMCLKm count clock signal can be calculated from the setting of the PWBAnBRSm register by the equation below.

  PWMCLKm count clock cycle = (PWBAnBRSm value × 2) × PCLK cycle

  In addition, PWBA can control operation when the on-chip debugger is in use by using the PWBAnEMU register.

#### PWGA

PWGA outputs PWM waveforms and A/D conversion trigger to PWSA by using the clock PWMCLKm input from PWBA.

- PWM waveform output PWGA_TOUTn
  This generator outputs PWM waveforms from the PWGA_TOUTn pin. The PWM cycle is the full count cycle of the PWGAnCNT register (12-bit free-running counter). Set the high-level period of PWM output in the PWGAnCSDR and PWGAnCRDR registers.

  The PWM waveform cycle and duty can be calculated by the equations below.

  PWM waveform cycle = PWGAnCNT (12-bit full count: $FFF_H$ + 1)
  × Count clock cycle
  = 4096 × PWMCLKm count clock cycle

  When PWGAnCRDR[11:0] > PWGAnCSDR[11:0],

  High-level period of PWM waveform =
  (PWGAnCRDR register value – PWGAnCSDR register value)
  × PWMCLKm count clock cycle

  PWM waveform duty (%) = High-level period of PWM waveform/PWM waveform cycle × 100 = (PWGAnCRDR register value – PWGAnCSDR register value)/4096 × 100

  Note that the PWM output is fixed to the low level when the PWGAnCRDR register value is equal to the PWGAnCSDR register value.

  When $1xxx_H$ is set in the PWGAnCRDR register (i.e. bit 12 is set to 1), the PWM output is fixed to the high level.

- A/D conversion trigger output PWGA_TRGOUTn
  The A/D conversion trigger signal PWGA_TRGOUTn for PWSA is generated when the PWGAnCTDR register value and the PWGAnCNT register value match while the PWM output PWGA_TOUTn is at the high level.
  The timing can be calculated by the equation below.

  A/D conversion trigger signal generation timing = PWGAnCTDR register value
  × PWMCLKm count clock cycle

- PWGA interrupt request signal PWGA_INTn
  PWGA generates the interrupt request signal PWGA_INTn at the falling edge of the PWM output PWGA_TOUTn.
  When the PWM output is fixed to the low level, PWGA_INTn is generated when the PWGAnCRDR register value and the PWGAnCNT register value (free-running counter value) match; when the PWM output is fixed to the high level, PWGA_INTn is generated at the timing of overflow of the PWGAnCNT register.

### PWSA

PWSA transmits the required setting information to the A/D converter and outputs the A/D conversion start trigger, based on the A/D conversion trigger signal PWGA_TRGOUTn from the PWM generator (PWGA).

- A/D conversion control by PWSA
  PWSA outputs the information required for the A/D conversion, which is set in the corresponding PWSAnPVCRx_y register for the channel number of the trigger input from PWGAn, (i.e., information on ADC physical channel, external MPX control, and error detection level selection) to the A/D converter.
  At the same timing, A/D conversion trigger (PWSA_ADTRG) is output to ADCA0 or ADCA1.
  (A maximum of eight input trigger signal PWGA_TRGOUTn data received during A/D conversion are stored and kept in PWSAnQUE.)
  The setting information to be output to the A/D converter is kept until the next trigger is generated.
  When the A/D conversion triggered by the PWM-Diag function is completed in the A/D converter, PWSA triggers the next A/D conversion based on the data stored in the PWSAnQUE register.
- Queuing of A/D conversion triggers from PWGA
  The A/D conversion trigger signal (PWGA_TRGOUTn) input from PWGAn is stored in the PWSAnQUEj register as a channel number. The PWSAnQUEj register stores a maximum of eight channel numbers of the A/D conversion trigger signal PWGA_TRGOUTn received during A/D conversion in a queue structure.
  A PWSA queue full interrupt occurs in the following states, when the queue of the PWSAnQUEj register becomes full
  - A trigger number is written to PWSAnQUE7
  - A trigger number has already been written to PWSAnQUE7 and cannot be written when PWGA_TRGOUTn is input.

### ADCA

A/D conversion is executed upon receipt of information required for A/D conversion and A/D conversion trigger from PWSA.

A/D conversion is executed using the PWM-Diag-dedicated scan group; on completion of the A/D conversion, it is reported to the PWSA.

For the basic operation of the A/D converter, see Section 31, A/D Converter (ADCA).

For the A/D converter operation with the PWM-Diag function, see Section 31.4.7.1, A/D Conversion with PWM-Diag Enabled.

### 30.1.9 Block Diagram

The following figure shows an example of connecting the LED control circuit combining the PWM-Diag and the A/D converter.

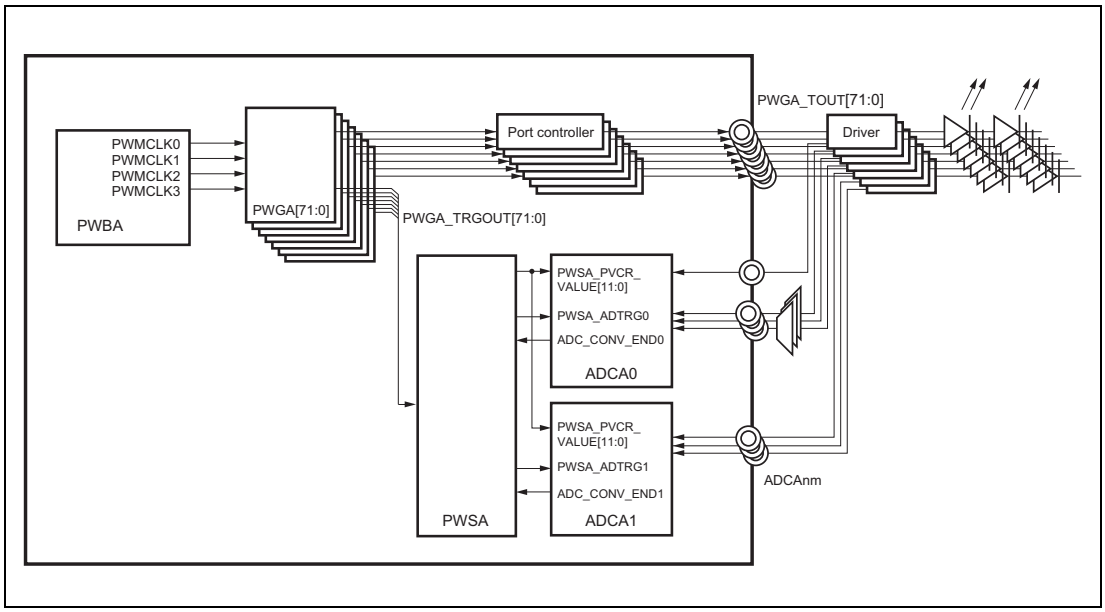



**Figure 30.1 Example of Connecting the LED Control Circuit using the PWM-Diag and the A/D Converter**

## 30.2 Registers

### 30.2.1 List of Registers

PWM Output/Diagnostic registers are listed in the following table.

<PWBAn_base>, <PWSAn_base>, and <PWGAn_base> are defined in **Section 30.1.2, Register Base Address**.

**Table 30.10 List of PWM Output/Diagnostic Registers**

| Module name | Register name | Symbol | Address |
|---|---|---|---|
| PWBAn | PWMCLKm cycle configuration register | PWBAnBRSm | <PWBAn_base> + $0004_H$ × m |
| | PWMCLKm enable status register | PWBAnTE | <PWBAn_base> + $0010_H$ |
| | PWMCLKm start trigger register | PWBAnTS | <PWBAn_base> + $0014_H$ |
| | PWMCLKm stop trigger register | PWBAnTT | <PWBAn_base> + $0018_H$ |
| | PWBA emulation register | PWBAnEMU | <PWBAn_base> + $001C_H$ |
| PWGAn | PWM output set condition register | PWGAnCSDR | <PWGAn_base> + $0000_H$ |
| | PWM output reset condition register | PWGAnCRDR | <PWGAn_base> + $0004_H$ |
| | PWGA_TRGOUTn generation condition register | PWGAnCTDR | <PWGAn_base> + $0008_H$ |
| | Buffer register reload trigger register | PWGAnRDT | <PWGAn_base> + $000C_H$ |
| | Buffer register reload status register | PWGAnRSF | <PWGAn_base> + $0010_H$ |
| | PWM cycle count register | PWGAnCNT | <PWGAn_base> + $0014_H$ |
| | PWGA control register | PWGAnCTL | <PWGAn_base> + $0020_H$ |
| | PWGAnCSDR buffer register | PWGAnCSBR | <PWGAn_base> + $0024_H$ |
| | PWGAnCRDR buffer register | PWGAnCRBR | <PWGAn_base> + $0028_H$ |
| | PWGAnCTDR buffer register | PWGAnCTBR | <PWGAn_base> + $002C_H$ |
| SLPWG | PWGA synchronous trigger register | SLPWGAk | <SLPW_base> + k × $4_H$ |
| PWSAn | PWSA control register | PWSAnCTL | <PWSAn_base> + $0000_H$ |
| | Trigger queue status register | PWSAnSTR | <PWSAn_base> + $0004_H$ |
| | Trigger queue status clear register | PWSAnSTC | <PWSAn_base> + $0008_H$ |
| | Trigger queue register | PWSAnQUEj | <PWSAn_base> + $0020_H$ + j × $4_H$ |
| | PWM-Diag mode A/D setting register | PWSAnPVCRx_y | <PWSAn_base> + $0040_H$ + x × $2_H$ |
| | PWSA emulation control register | PWSAnEMU | <PWSAn_base> + $000C_H$ |

### 30.2.1.1 PWBAnBRSm Register

This register sets the clock cycle of PWMCLKm.

**Access:** This register can be read or written in 16-bit units.

**Address:** <PWBAn_base> + $0004_H$ × m

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | PWBAnBRSm[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 30.11 PWBAnBRSm Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 11 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 10 to 0 | PWBAnBRSm [10:0] | Register for setting the clock cycle of PWMCLKm.<br>– PWBAnBRSm = 0: PWMCLKm = PCLK<br>– PWBAnBRSm = 1: PWMCLKm = PCLK / 2 × 1<br>– PWBAnBRSm = 2: PWMCLKm = PCLK / 2 × 2<br>...<br>– PWBAnBRSm = n: PWMCLKm = PCLK / 2 × n (n = 1 to 2047)<br>These bits can only be rewritten when all counters using PWMCLKm are stopped (PWBAnTE.PWBATEm = 0). |

### 30.2.1.2 PWBAnTE Register

This is a status register that indicates the output status of PWMCLKm (m = 0 to 3).

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <PWBAn_base> + $0010_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | PWBAnTE3 | PWBAnTE2 | PWBAnTE1 | PWBAnTE0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 30.12 PWBAnTE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned. |
| 3 | PWBAnTE3 | A status flag indicating the operation status of PWMCLK3<br>0: Not operating<br>1: Operating |
| 2 | PWBAnTE2 | A status flag indicating the operation status of PWMCLK2<br>0: Not operating<br>1: Operating |
| 1 | PWBAnTE1 | A status flag indicating the operation status of PWMCLK1<br>0: Not operating<br>1: Operating |
| 0 | PWBAnTE0 | A status flag indicating the operation status of PWMCLK0<br>0: Not operating<br>1: Operating |

### 30.2.1.3 PWBAnTS Register

This register is a start trigger register for PWMCLKm (m = 0 to 3).

**Access:** This register is a write-only register that can be written in 8-bit units.

**Address:** <PWBAn_base> + $0014_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | PWBAnTS3 | PWBAnTS2 | PWBAnTS1 | PWBAnTS0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | W | W | W | W |

**Table 30.13 PWBAnTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When writing, write the value after reset. |
| 3 | PWBAnTS3 | Start Trigger for PWMCLK3<br>0: Writing 0 has no effect.<br>1: Starts the output of PWMCLK3. |
| 2 | PWBAnTS2 | Start Trigger for PWMCLK2<br>0: Writing 0 has no effect.<br>1: Starts the output of PWMCLK2. |
| 1 | PWBAnTS1 | Start Trigger for PWMCLK1<br>0: Writing 0 has no effect.<br>1: Starts the output of PWMCLK1. |
| 0 | PWBAnTS0 | Start Trigger for PWMCLK0<br>0: Writing 0 has no effect.<br>1: Starts the output of PWMCLK0. |

### 30.2.1.4 PWBAnTT Register

This register is a stop trigger register for PWMCLKm (m = 0 to 3).

**Access:** This register is a write-only register that can be written in 8-bit units.

**Address:** <PWBAn_base> + $0018_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | PWBAnTT3 | PWBAnTT2 | PWBAnTT1 | PWBAnTT0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | W | W | W | W |

**Table 30.14 PWBAnTT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When writing, write the value after reset. |
| 3 | PWBAnTT3 | Stop Trigger for PWMCLK3<br>0: Writing 0 has no effect.<br>1: Stops the output of PWMCLK3. |
| 2 | PWBAnTT2 | Stop Trigger for PWMCLK2<br>0: Writing 0 has no effect.<br>1: Stops the output of PWMCLK2. |
| 1 | PWBAnTT1 | Stop Trigger for PWMCLK1<br>0: Writing 0 has no effect.<br>1: Stops the output of PWMCLK1. |
| 0 | PWBAnTT0 | Stop Trigger for PWMCLK0<br>0: Writing 0 has no effect.<br>1: Stops the output of PWMCLK0. |

### 30.2.1.5 PWBAnEMU Register

This register sets the operation during emulation.

**Access:** This register can be read or written in 8-bit units.

**Address:** <PWBAn_base> + $001C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PWBAnSVSDIS | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R |

**Table 30.15 PWBAnEMU Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | PWBAnSVSDIS | (When the EPC.SVSTOP bit = 0)<br>The count clock is provided continuously when the debugger is controlling the microcontroller (by using break points, etc.), regardless of the value of this bit (1 or 0).<br><br>(When the EPC.SVSTOP bit = 1)<br>0: The count clock is stopped when the debugger is controlling the microcontroller (by using break points, etc.).<br>1: The count clock is provided continuously when the debugger is controlling the microcontroller (by using break points, etc.).<br>This bit can only be rewritten when all counters using PWMCLKm are stopped (PWBAnTE.PWBATEm = 0). |
| 6 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

### 30.2.1.6 PWGAnCTL — PWGA Control Register

PWGAnCTL is used to select the count clock from PWBA.

**Access:** This register can be read or written in 8-bit units.

**Address:** <PWGAn_base> + $0020_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | PWGAnCKS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 30.16 PWGAnCTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | PWGAnCKS [1:0] | Count Clock Enable Input PWMCLK3 to PWMCLK0 Select<br>00: Uses PWMCLK0 as count clock<br>01: Uses PWMCLK1 as count clock<br>10: Uses PWMCLK2 as count clock<br>11: Uses PWMCLK3 as count clock<br>These bits can only be rewritten when the PWGAn operation is stopped (SLPWGAk.SLPWGA[31:0] = 0). |

### 30.2.1.7 PWGAnCNT — PWM Cycle Count Register

This is a count register.

**Access:** This register is a read-only register that can be read in 16-bit units.

**Address:** <PWGAn_base> + $0014_H$

**Value after reset:** $0FFF_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | PWGAnCNT[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 30.17 PWGAnCNT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 12 | Reserved | When read, the value after reset is returned. |
| 11 to 0 | PWGAnCNT [11:0] | 12-bit counter value |

### 30.2.1.8 PWGAnCSDR — PWM Output Set Condition Register

This register sets the setting condition for PWGA_TOUTn output.

**Access:** This register can be read or written in 16-bit units.

**Address:** <PWGAn_base> + $0000_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | PWGAnCSDR[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 30.18 PWGAnCSDR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 12 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 11 to 0 | PWGAnCSDR [11:0] | These bits set the setting condition for PWM output.<br>The set value is reflected to the PWGAnCSBR register at the start of PWGAn operation (SLPWGAk.SLPWGA of the corresponding CH = 1) or when a simultaneous rewrite is performed (PWGAnRDT.PWGAnRDT = 1). |

### 30.2.1.9 PWGAnCRDR — PWM Output Reset Condition Register

This register sets the reset condition for PWGA_TOUTn output.

**Access:** This register can be read or written in 16-bit units.

**Address:** <PWGAn_base> + $0004_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | PWGAnCRDR[12:0] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 30.19 PWGAnCRDR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 to 0 | PWGAnCRDR [12:0] | These bits set the reset condition for PWM output.<br>The set value is reflected to the PWGAnCRBR register at the start of PWGAn operation (SLPWGAk.SLPWGA of the corresponding CH = 1) or when a simultaneous rewrite is performed (PWGAnRDT.PWGAnRDT = 1). |

### 30.2.1.10 PWGAnCTDR — PWGA_TRGOUTn Generation Condition Register

This register sets the generation condition for PWGA_TRGOUTn.

**Access:** This register can be read or written in 16-bit units.

**Address:** <PWGAn_base> + $0008_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | PWGAnCTDR[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 30.20 PWGAnCTDR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 12 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 11 to 0 | PWGAnCTDR [11:0] | These bits set the A/D conversion trigger generation condition for PWSAn. The set value is reflected to the PWGAnCTBR register at the start of PWGAn operation (SLPWGAk.SLPWGA of the corresponding CH = 1) or when a simultaneous rewrite is performed (PWGAnRDT.PWGAnRDT = 1). |

### 30.2.1.11 PWGAnCSBR — PWGAnCSDR Buffer Register

This is a buffer register for the PWGAnCSDR register.

**Access:** This register is a read-only register that can be read in 16-bit units.

**Address:** <PWGAn_base> + $0024_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | PWGAnCSBR[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 30.21 PWGAnCSBR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 12 | Reserved | When read, the value after reset is returned. |
| 11 to 0 | PWGAnCSBR [11:0] | The set value is reflected to the PWGAnCSDR register at the start of PWGAn operation (SLPWGAk.SLPWGA of the corresponding CH = 1) or when a simultaneous rewrite is performed (PWGAnRDT.PWGAnRDT = 1). When the value matches the PWGAnCNT register value, the pin output is driven high. |

### 30.2.1.12 PWGAnCRBR — PWGAnCRDR Buffer Register

This is a buffer register for the PWGA_TOUTn reset condition.

**Access:** This register is a read-only register that can be read in 16-bit units.

**Address:** <PWGAn_base> + $0028_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | PWGAnCRBR[12:0] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 30.22 PWGAnCRBR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 13 | Reserved | When read, the value after reset is returned. |
| 12 to 0 | PWGAnCRBR [12:0] | The set value is reflected to the PWGAnCRDR register at the start of PWGAn operation (SLPWGAk.SLPWGA of the corresponding CH = 1) or when a simultaneous rewrite is performed (PWGAnRDT.PWGAnRDT = 1).<br>When the value matches the PWGAnCNT register value, the pin output is driven low. |

### 30.2.1.13 PWGAnCTBR — PWGAnCTDR Buffer Register

This is a buffer register for the PWGA_TRGOUTn generation condition.

**Access:** This register is a read-only register that can be read in 16-bit units.

**Address:** <PWGAn_base> + $002C_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | PWGAnCTBR[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 30.23 PWGAnCTBR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 12 | Reserved | When read, the value after reset is returned. |
| 11 to 0 | PWGAnCTBR [11:0] | The set value is reflected to the PWGAnCTDR register at the start of PWGAn operation (SLPWGAk.SLPWGA of the corresponding CH = 1) or when a simultaneous rewrite is performed (PWGAnRDT.PWGAnRDT = 1).<br>When the value matches the PWGAnCNT register value, a trigger is transmitted to PWSAn. |

### 30.2.1.14 PWGAnRSF — Buffer Register Reload Status Register

This register is a status register for simultaneous rewrite control.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <PWGAn_base> + $0010_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | PWGAnRSF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 30.24 PWGAnRSF Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | PWGAnRSF | Simultaneous Rewrite Control Status<br>0: Simultaneous rewrite is enabled. This value indicates the completion of simultaneous rewrite after the generation of a simultaneous rewrite trigger signal.<br>1: Simultaneous rewrite is in progress. This value indicates the waiting state for completion. |

### 30.2.1.15 PWGAnRDT — Buffer Register Reload Trigger Register

This is a simultaneous rewrite request trigger register.

**Access:** This register is a write-only register that can be written in 8-bit units.

**Address:** <PWGAn_base> + $000C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | PWGAnRDT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 30.25 PWGAnRDT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | PWGAnRDT | Simultaneous Rewrite Request Trigger<br>0: Writing 0 has no effect.<br>1: Triggers the simultaneous rewrite request for the compare registers (PWGAnCSDR, PWGAnCRDR, and PWGAnCTDR), and sets PWGAnRSF.PWGAnRSF to 1. |

### 30.2.1.16 SLPWGAk — PWGA Synchronous Trigger Register (k = 0 to 2)

This register triggers start and stop for multiple channels simultaneously.

**Access:** This register can be read or written in 32-bit units.

**Address:** <SLPW_base> + k × $4_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SLPWGA[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SLPWGA[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 30.26 SLPWGAk Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | SLPWGA [31:0] | Trigger start and stop to multiple channels simultaneously.<br>0: Stops the corresponding channels.<br>1: Starts the corresponding channels.<br>The bits correspond to the following channels.<br>SLPWGA0.SLPWGA[31:0]: PWGA31 - PWGA0<br>SLPWGA1.SLPWGA[31:0]: PWGA63 - PWGA32<br>SLPWGA2.SLPWGA[7:0]: PWGA71 - PWGA64 |

### 30.2.1.17 PWSAnCTL Register

This register is used to control operations of PWSA.

**Access:** This register can be read or written in 8-bit units.

**Address:** <PWSAn_base> + $0000_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | PWSAnENBL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 30.27 PWSAnCTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | PWSAnENBL | Operation Permission Control<br>0: Operation is prohibited (initial state).<br>Writing 0 initializes PWSAnSTR and PWSAnQUEj.<br>1: Operation is enabled. |

### 30.2.1.18 PWSAnSTR Register

This is a status register that indicates whether the number of a channel for which an A/D conversion trigger has been generated is stored in a PWSAnQUEj register.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <PWSAn_base> + $0004_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | PWSAnQFL | PWSAnQNE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 30.28 PWSAnSTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. |
| 1 | PWSAnQFL | Indicates the queuing state of the A/D conversion trigger.<br>0: Some PWSAnQUEj registers do not store a channel number.<br>1: All of the PWSAnQUEj registers store a channel number. |
| 0 | PWSAnQNE | Bit indicating that there is a trigger in the trigger queue<br>0: A channel number is not stored in a PWSAnQUEj register, or A/D conversion is in progress while only PWSAnQUE0 stores a channel number.<br>1: The number of the channel waiting for conversion is stored in j = 1 and subsequent PWSAnQUEj registers. |

### 30.2.1.19 PWSAnSTC Register

This register clears the status of the PWSAnSTR register.

**Access:** This register is a write-only register that can be written in 8-bit units.

**Address:** <PWSAn_base> + $0008_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | PWSAnCLFL | PWSAnCLNE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | W | W |

**Table 30.29 PWSAnSTC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When writing, write the value after reset. |
| 1 | PWSAnCLFL | PWSAnQFL Clear<br>0: PWSAnQFL retains the status (Writing 0 has no effect).<br>1: PWSAnQFL is cleared to 0. |
| 0 | PWSAnCLNE | PWSAnQNE Clear<br>0: PWSAnQNE retains the status (Writing 0 has no effect).<br>1: PWSAnQNE is cleared to 0. |

### 30.2.1.20 PWSAnQUEj (j = 0 to 7) Register

This register stores the channel number that received the trigger from PWGAn.

**Access:** This register is a read-only register that can be read in 8-bit units.

**Address:** <PWSAn_base> + $0020_H + j \times 4_H$

**Value after reset:** $7F_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | PWSAnQUEj[6:0] | | | | | | |
| Value after reset | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R |

**Table 30.30 PWSAnQUEj Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned. |
| 6 to 0 | PWSAnQUEj [6:0] | These bits hold the channel number (0 to 71) of the PWGA for which a trigger was generated in order from PWSAnQUE0 to PWSAnQUE7.<br>After the A/D conversion of PWSAnQUE0 is completed, the values in PWSAnQUE1 to PWSAnQUE7 shift to PWSAnQUE0 to PWSAnQUE6. |

**NOTE**

If a trigger occurs simultaneously for multiple channels, the trigger with the smaller channel number has priority.

### 30.2.1.21 PWSAnPVCRx_y (x = 00, 02, 04 ... 70, y = 01, 03, 05 ... 71) Register

This register is used to set the corresponding A/D converter for each channel.

Two consecutive channels are set such as PWSA0PVCR02_03, and the 16 higher-order bits of each register correspond to an odd-numbered channel while the 16 lower-order bits correspond to an even-numbered channel.

At the generation of a trigger, the set value is transmitted to the ADCAnPWDVCR register of the A/D converter.

For the ADCAnPWDVCR register, see Section 31, A/D Converter (ADCA).

**Access:** This register can be read or written in 32-bit units.

**Address:** <PWSAn_base> + $0040_H$ + x × $2_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | PWSAnSLADy | PWSAnVRDTy[27] | PWSAnVRDTy[26:24] | | | PWSAnVRDTy[23:22] | | PWSAnVRDTy[21:16] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | PWSAnSLADx | PWSAnVRDTx[11] | PWSAnVRDTx[10:8] | | | PWSAnVRDTx[7:6] | | PWSAnVRDTx[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 30.31 PWSAnPVCRx_y Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 28 | PWSAnSLADy | RH850/F1K 100 pin products:<br>When writing, write the value after reset.<br>RH850/F1K 144 and 176 pin products:<br>ADCA Select (odd-numbered channel)<br>0: Output to ADCA0.<br>1: Output to ADCA1. |
| 27 | PWSAnVRDTy[27] | This bit indicates the set value of the ADCAnPWDVCR.MPXE bit (odd-numbered channel). |
| 26 to 24 | PWSAnVRDTy[26:24] | These bits indicate the set value of the ADCAnPWDVCR.MPXV[2:0] bits (odd-numbered channel). |
| 23, 22 | PWSAnVRDTy[23:22] | These bits indicate the set value of the ADCAnPWDVCR.ULS[1:0] bits (odd-numbered channel). |
| 21 to 16 | PWSAnVRDTy[21:16] | These bits indicate the set value of the ADCAnPWDVCR.GCTRL[5:0] bits (odd-numbered channel). |
| 15 to 13 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 12 | PWSAnSLADx | RH850/F1K 100 pin products:<br>When writing, write the value after reset.<br><br>RH850/F1K 144 and 176 pin products:<br>ADCA Select (even-numbered channel)<br>0: Output to ADCA0.<br>1: Output to ADCA1. |
| 11 | PWSAnVRDTx[11] | This bit indicates the set value of the ADCAnPWDVCR.MPXE bit. (even-numbered channel) |

**Table 30.31 PWSAnPVCRx_y Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 10 to 8 | PWSAnVRDTx [10:8] | These bits indicate the set value of the ADCAnPWDVCR.MPXV[2:0] bits. (even-numbered channel) |
| 7, 6 | PWSAnVRDTx [7:6] | These bits indicate the set value of the ADCAnPWDVCR.ULS[1:0] bits. (even-numbered channel) |
| 5 to 0 | PWSAnVRDTx [5:0] | These bits indicate the set value of the ADCAnPWDVCR.GCTRL[5:0] bits. (even-numbered channel) |

### 30.2.1.22 PWSAnEMU — Emulation Control Register

This register is used to set the operation for emulation.

**Access:** This register can be read or written in 8-bit units.

**Address:** <PWSAn_base> + $000C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | PWSAnSVSDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 30.32 PWSAnEMU Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | PWSAnSVSDIS | (When the EPC.SVSTOP bit = 0)<br>The operation continues when the debugger is controlling the microcontroller (by using break points, etc.), regardless of the value of this bit (1 or 0).<br><br>(When the EPC.SVSTOP bit = 1)<br>0: When the debugger is controlling the microcontroller (by using break points, etc.);<br>– The output state to A/D is retained, the ADC_CONV_ENDn input at a break point is internally retained, and PWSAnQUEj is updated after break release.<br>– The PWGA_TRGOUTn input is accepted even at a break, and PWSA_INT_QFULL is also output.<br>– Reading and writing to the register is possible.<br>1: The operation continues when the debugger is controlling the microcontroller (by using break points, etc.).<br><br>The above bit can only be rewritten when all counters using PWMCLKm are stopped (PWBAnTE.PWBATEm = 0), the operation of all channels PWGAn has stopped (SLPWGAk.SLPWGA), and no trigger has been generated from any of the channels PWGAn (PWSAnQUE0 is the value after reset). |

## 30.3 Operating Procedure

Procedures for setting when starting and stopping operation of PWM-Diag are illustrated below.

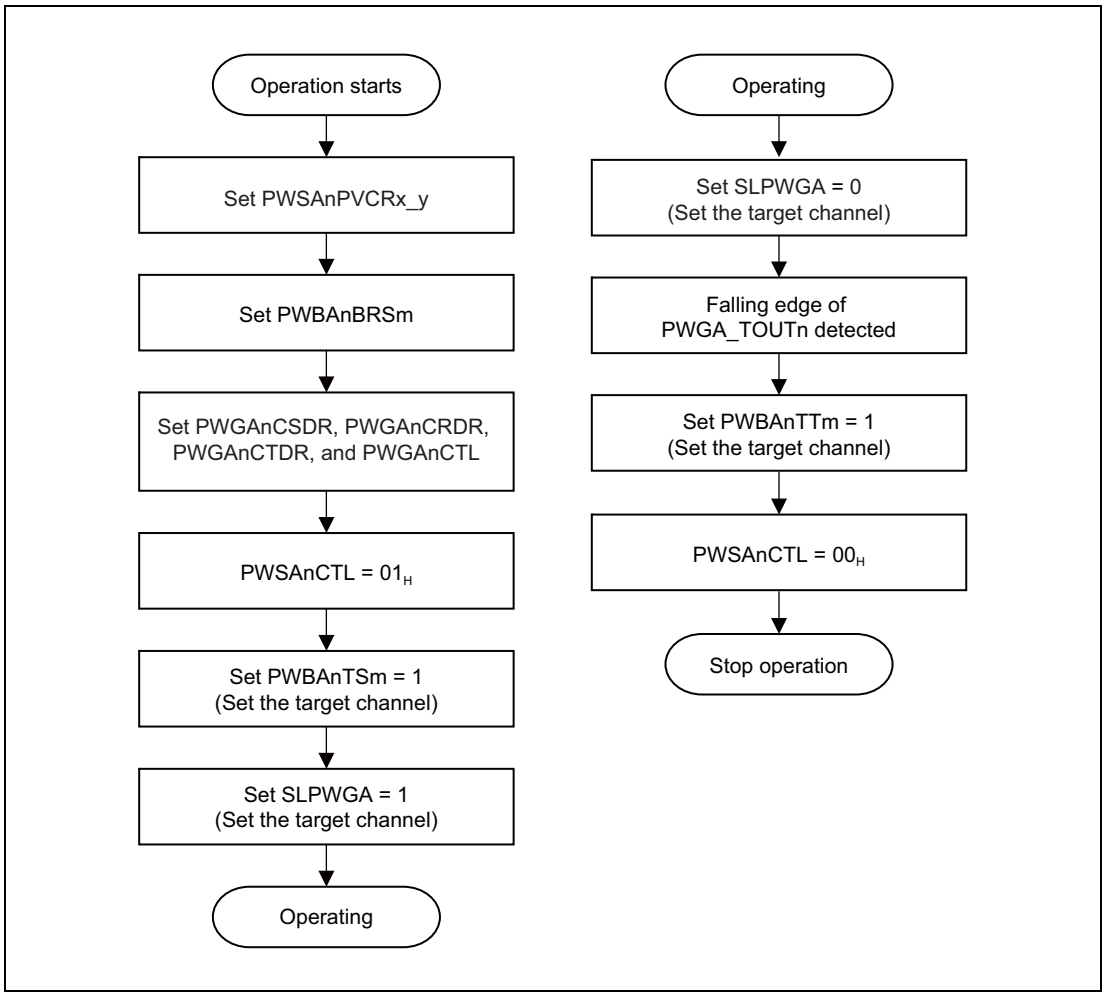
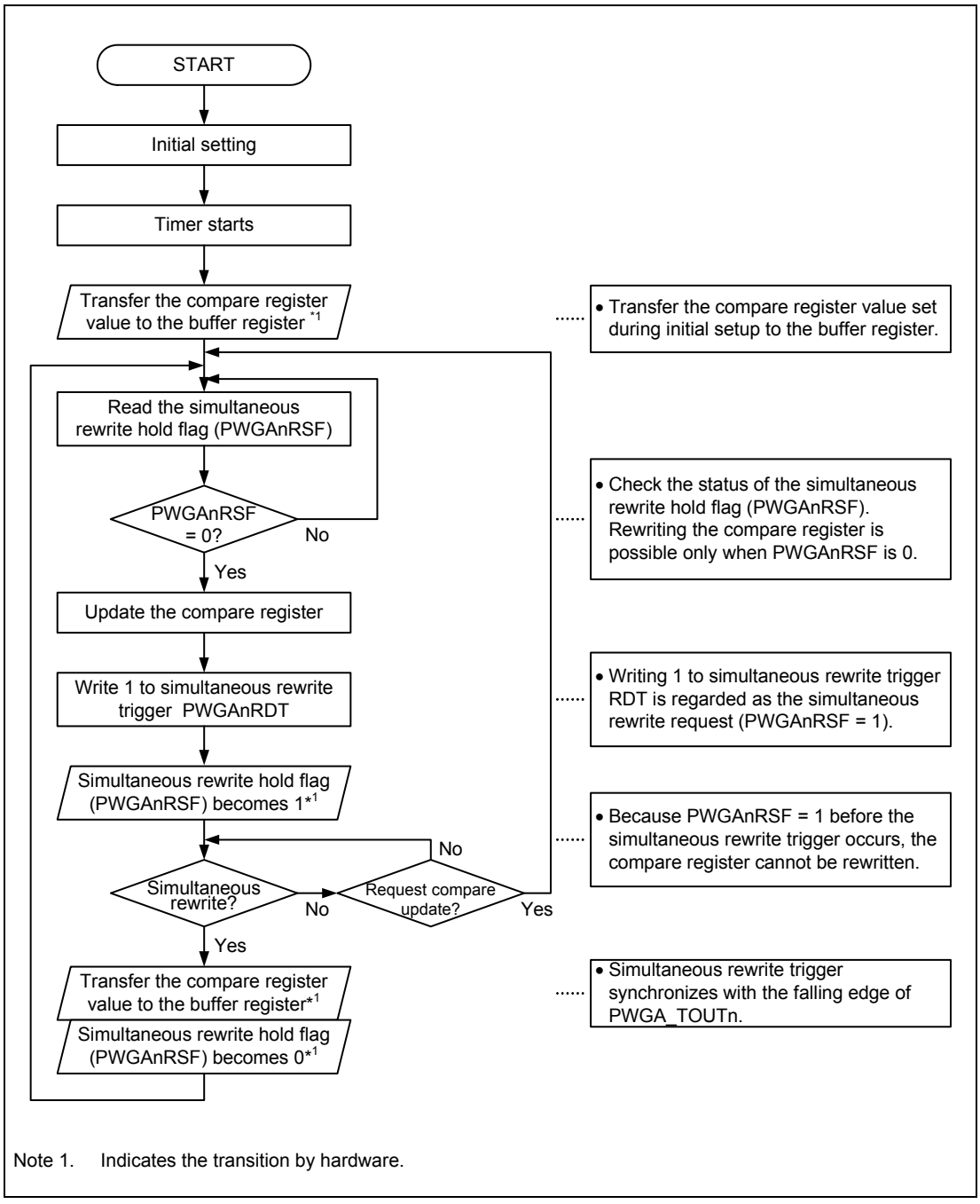



**Figure 30.2 PWM-Diag Operating Procedure**

Procedures for simultaneous rewrite of PWGA are illustrated below.

The described term “compare register” indicates PWGAnCSDR, PWGAnCRDR, or PWGAnCTDR.

In addition, the described term “buffer register” indicates PWGAnCSBR, PWGAnCRBR, or PWGAnCTBR.




Note 1. Indicates the transition by hardware.

**Figure 30.3 Simultaneous Rewrite Procedure**

# 30.4 Operation Waveform of PWM-Diag

## 30.4.1 PWM Waveform Output by PWGA and Operation Waveform for A/D Conversion Trigger Output

### 30.4.1.1 Basic Operation Waveform of PWGA

The basic operation waveforms of PWGA are illustrated below.

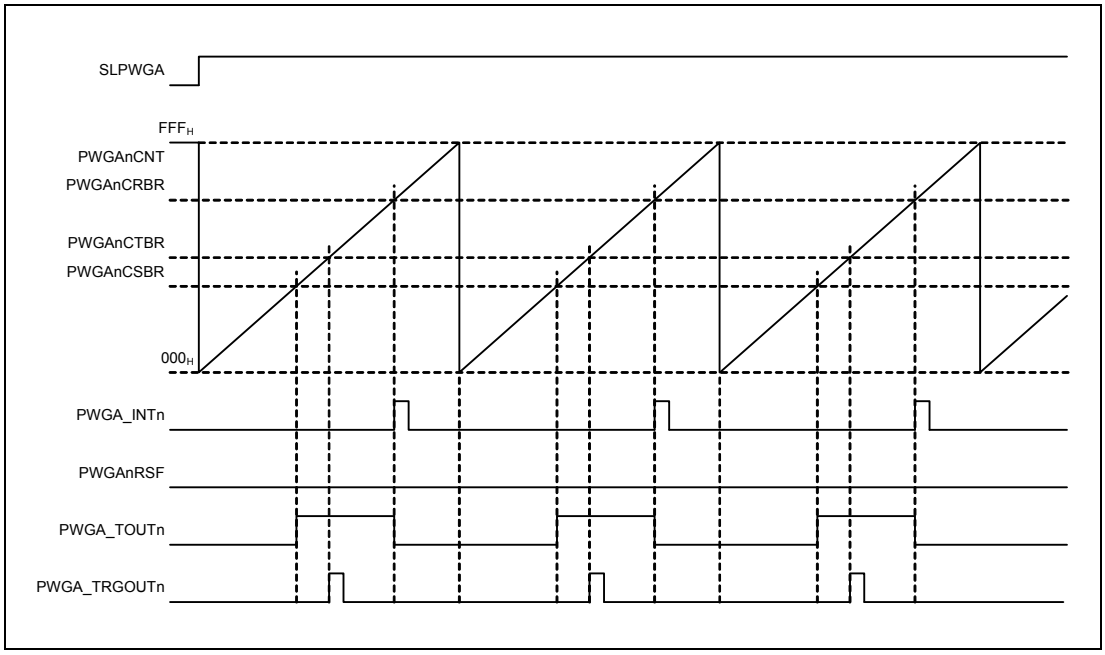



**Figure 30.4 Basic Waveform**

### 30.4.1.2 Operation Waveform when Simultaneous Rewrite for PWGA is Executed

The following figure illustrates the operation waveforms when simultaneous rewrite for PWGA is executed.

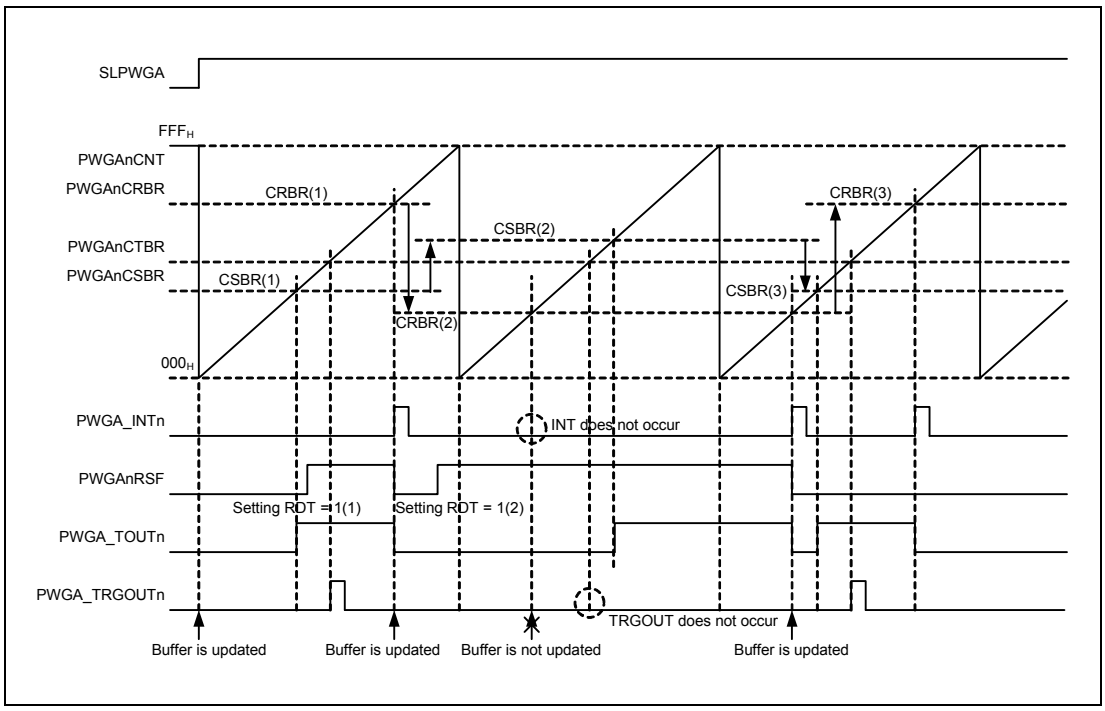



**Figure 30.5 Waveform when Simultaneous Rewrite is Executed**

Simultaneous rewrite is executed by re-setting the PWGAnCSDR and PWGAnCRDR registers, then setting either the PWGAnRDT or SLPWGAk register.

Moreover, if the relationship between set values in one interval is PWGAnCSDR > PWGAnCRDR, a falling edge in that interval is meaningless, and the falling edge in the next interval is valid.

Moreover, PWGA_TRGOUTn does not become valid unless PWGA_TOUTn is at the high level.

### 30.4.1.3 Operation Waveform when Stopping and Restarting PWGA Operation

The following figure illustrates the operation waveforms when stopping and restarting PWGA operation.

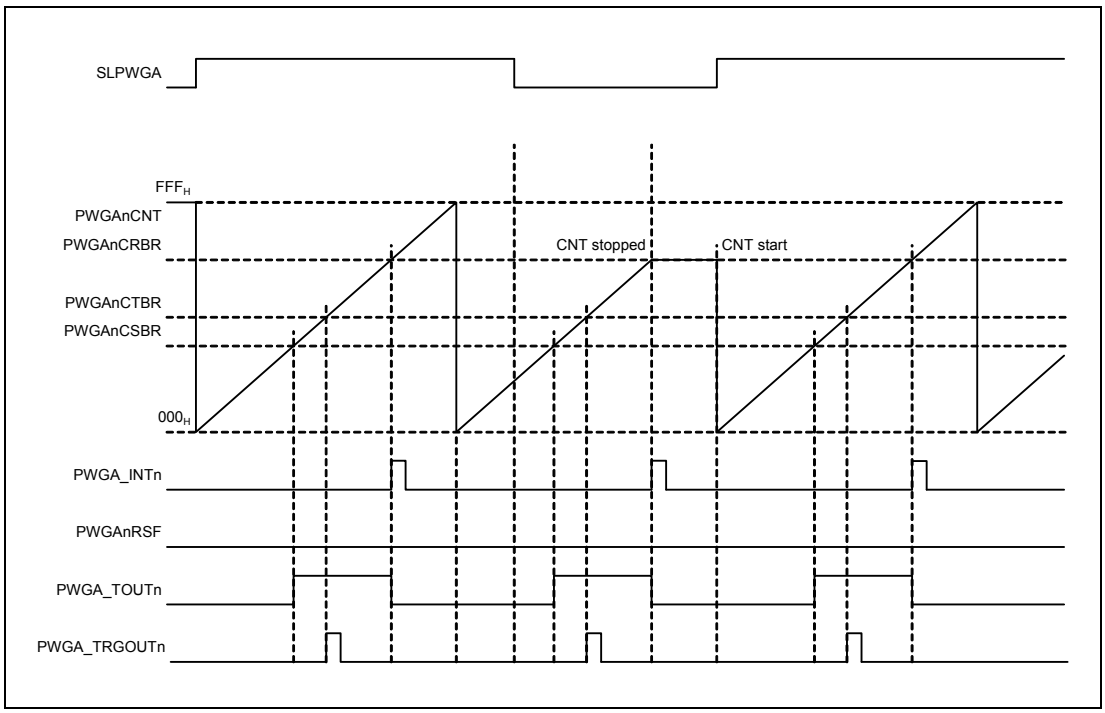



**Figure 30.6 Stopping and Resuming Operation (1)**

After the setting of SLPWGA has been changed from 1 to 0, PWGAnCNT stops operation because PWGA_INTn is generated.

After PWGA_INTn has been generated, by changing the setting of SLPWGA from 0 to 1, PWGAnCNT resumes counting from $000_H$.

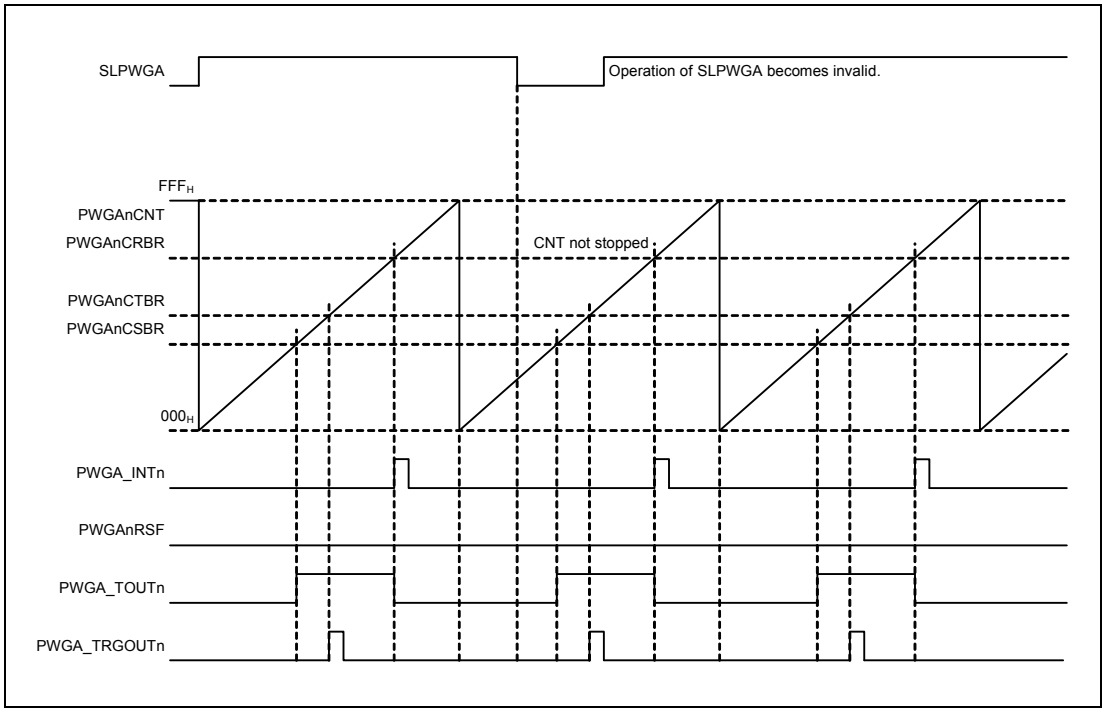



**Figure 30.7 Stopping and Resuming Operation (2)**

After the setting of SLPWGA has been changed from 1 to 0, if the setting of SLPWGA is changed from 0 to 1 before PWGA_INTn is generated, operations of SLPWGA become invalid, and PWGAnCNT continues counting.

### 30.4.1.4 Waveforms of PWGA Operation with Specific Settings

The following figures illustrate the waveforms of PWGA operation with specific settings.

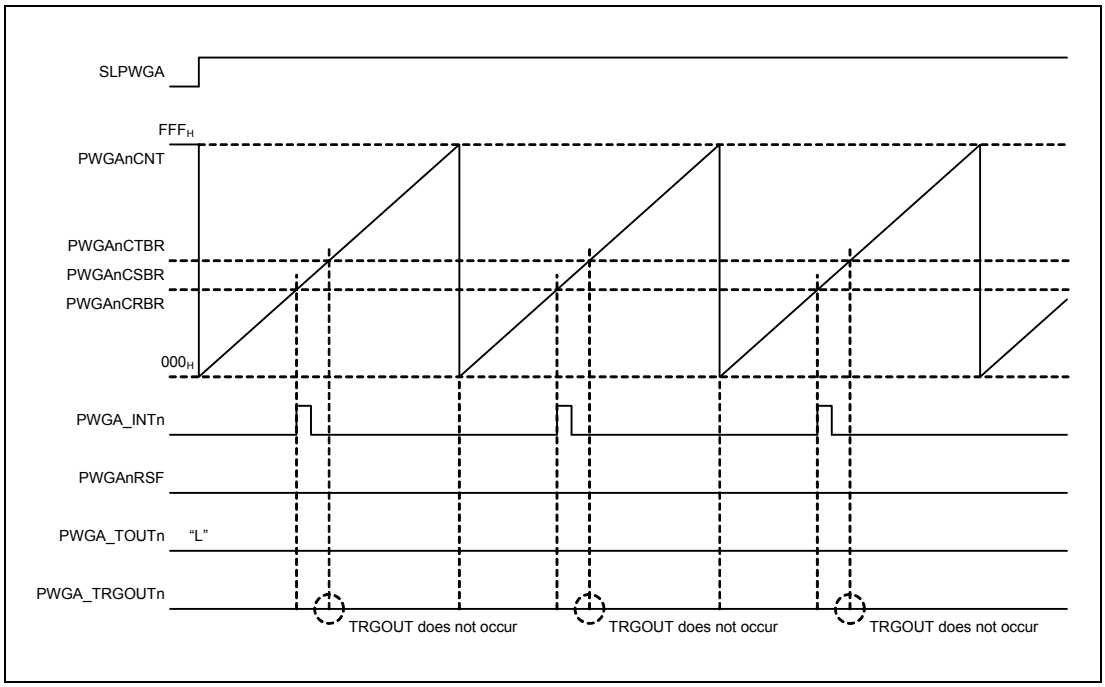



**Figure 30.8 PWGA_TOUTn = 0% Output Waveform**

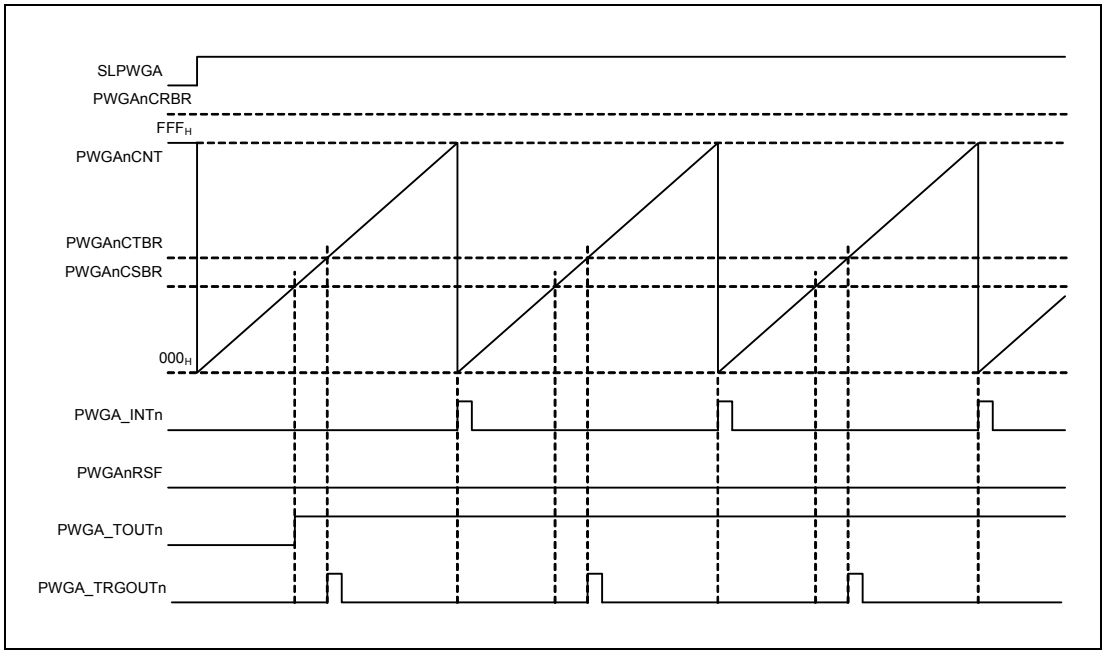
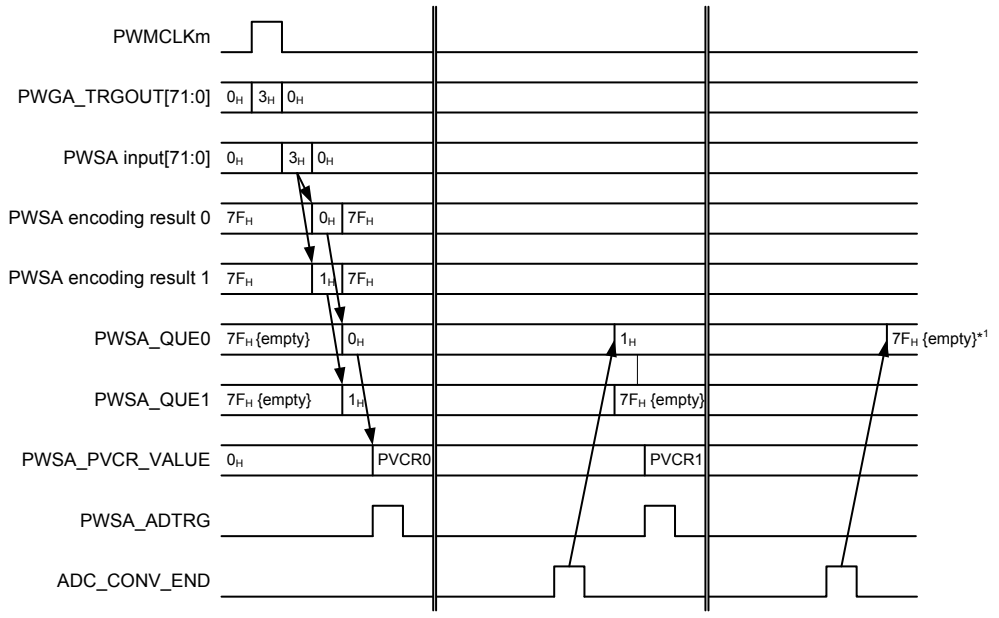



**Figure 30.9 PWGA_TOUTn = 100% Output Waveform**

### 30.4.2 Operation Waveform when A/D Conversion Trigger Occurs in PWSA

An example of the PWSA operation is shown below.




Note 1. QUE0 indicates the PWGA_TRG number while the A/D conversion is executed. The initial value (while the A/D conversion is not executed) is $7F_H$.

**Figure 30.10 Example of PWSA Operation**

(1) Triggers occur simultaneously in channels 0 and 1 of PWGA. Channel 0 with the smaller channel number is stored in PWSAnQUE0, and channel 1 with the larger channel number is stored in PWSAnQUE1. The lower 16-bit data of PWSAnPVCR00_01 corresponding to the value stored in PWSAnQUE0 is transmitted to the A/D converter and a trigger is output to the A/D converter. At this time, as the A/D conversion for channel 1 is in the waiting state, the PWSAnSTR.PWSAnQNE bit is set.

(2) On completion of A/D conversion executed in step (1), the channel number of PWSAnQUE1 shifts to PWSAnQUE0 and PWSAnQUE1 enters the empty state.
After that, as similar to step (1), the upper 16-bit data of PWSAnPVCR00_01 corresponding to the value stored in PWSAnQUE0 is transmitted to the A/D converter and a trigger is output to the A/D converter.

(3) On completion of A/D conversion executed in step (2), PWSAnQUE0 enters the empty state.

## 30.5 PWM-Diag Related Function in A/D Converter (ADCA)

This section describes the A/D converter used in the PWM-Diag function.

### 30.5.1 ADCA Registers when the PWM-Diag Function is Used

- Before starting PWSA operation, the A/D converter must be set using the following register.
  - PWM-Diag scan group control register (ADCAnPWDSGCR)
- When the PWM-Diag is running, the PWSAnPVCRx_y value corresponding to the channel under conversion is set in the following register of the A/D converter.
  - PWM-Diag virtual channel register (ADCAnPWDVCR)
- After completion of A/D conversion, the conversion result can be checked by reading the following registers.
  - PWM-Diag data register (ADCAnPWDTSNDR)
  - PWM-Diag data supplementary information register (ADCAnPWDDIR)
- When A/D conversion result is outside the expected range, it can be confirmed using the upper/lower limit error detection function. The upper/lower limit error detection function is set by the following register.
  - Upper limit/lower limit error register (ADCAnULER)
- The scan end flag of the PWM-Diag scan group can be cleared using the following register.
  - PWM-Diag scan end flag clear register (ADCAnPWDSGSEFCR)

# Section 31 A/D Converter (ADCA)

This section contains a generic description of the A/D Converter (ADCA).

The first part of this section describes all RH850/F1K specific properties, such as the number of units, register base addresses, etc. The remainder of the section describes the functions and registers of the ADCA.

## 31.1 Features of RH850/F1K ADCA

### 31.1.1 Number of Units and Channels

This microcontroller has the following number of ADCA units.

**Table 31.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of Units | 1 | 2 | 2 |
| Name | ADCAn (n = 0) | ADCAn (n = 0, 1) | ADCAn (n = 0, 1) |

An ADCAn unit has the same number of physical channels as the number of A/D input pins and the same number of virtual channels as the number of addresses where the results of A/D conversion will be stored. The numbers of channels on individual products are as listed below.

**Table 31.2 Unit Configurations and Physical Channels**

| Unit Name (Number of Channels) ADCAn | | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|---|
| ADCA0 | 12 bit pin for conversion*1 | 16 | 16 | 16 |
| | 10 bit pin for conversion*2 | 20 | 20 | 20 |
| ADCA1 | 12 bit pin for conversion*1 | — | 8 | 16 |
| | 10 bit pin for conversion*2 | — | 4 | 8 |

Note 1. When 10-bit mode is selected, this pin can be used for 10-bit conversion.

Note 2. When 12-bit mode is selected but a pin is for 10-bit conversion, the 2 low-order bits of the result of conversion must be masked before use.

**Table 31.3 Unit Configurations and Virtual Channels**

| Unit Name (Number of Channels) ADCAn | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| ADCA0 | 50 | 50 | 50 |
| ADCA1 | — | 36 | 36 |

Table 31.4 Indices

| Index | Description |
|---|---|
| n | Throughout this section, the individual ADCA units are identified by the index “n” (n = 0, 1); for example, ADCAnPWDVCR indicates the PWM-Diag virtual channel register. |
| m | Throughout this section, the individual physical channels (channels in the unit) of ADCAn are identified by the index “m”; for example, ANInm. |
| j | Throughout this section, the individual virtual channels of ADCAn are identified by the index “j”; for example, ADCAnVCRj indicates the virtual channel register. |
| x | Throughout this section, the individual scan groups (SG) of ADCAn are identified by the index “x” (x = 1 to 3); for example, ADCAnSGSTCRx indicates the scan group x start control register. |
| k | Throughout this section, the individual physical channel numbers for T&H are identified by the index “k” (k = 0 to 5); for example, THkE is T&H enable bit of the T&H enable register (ADCAnTHER). |

The following table shows values indicated by the indices of each product.

Table 31.5 Indices of Products

| Indices of Each Product | | |
|---|---|---|
| **RH850/F1K 100 pins** | **RH850/F1K 144 pins** | **RH850/F1K 176 pins** |
| m = 0 to 35 (ADCA0) | m = 0 to 35 (ADCA0)<br>m = 0 to 7 (ADCA1)<br>16 to 19 (ADCA1) | m = 0 to 35 (ADCA0)<br>m = 0 to 23 (ADCA1) |
| j = 00 to 49 (ADCA0) | j = 00 to 49 (ADCA0)<br>j = 00 to 35 (ADCA1) | j = 00 to 49 (ADCA0)<br>j = 00 to 35 (ADCA1) |
| x = 1 to 3 (ADCA0) | x = 1 to 3 (ADCA0)<br>x = 1 to 3 (ADCA1) | x = 1 to 3 (ADCA0)<br>x = 1 to 3 (ADCA1) |
| k = 0 to 5 (ADCA0) | k = 0 to 5 (ADCA0) | k = 0 to 5 (ADCA0) |

## 31.1.2 Register Base Address

ADCAn base addresses are listed in the following table.

ADCAn register addresses are given as offsets from the base addresses.

Table 31.6 Register Base Addresses

| Base Address Name | Base Address |
|---|---|
| <ADCA0_base> | FFF2 $0000_H$ |
| <ADCA1_base> | FFD6 $D000_H$ |

### 31.1.3 Clock Supply

The ADCAn clock supply is shown in the following table.

**Table 31.7 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| ADCA0 | ADCLK | CKSCLK_AADCA | Module clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CKSCLK_AADCA | |
| ADCA1 | ADCLK | CKSCLK_IADCA | Module clock |
| | Register access clock | CPUCLK2 | Bus clock |

### 31.1.4 Interrupt Requests

ADCAn interrupt requests are listed in the following table.

**Table 31.8 Interrupt Requests**

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number | Other Trigger Signals |
|---|---|---|---|---|
| **ADCA0** | | | | |
| INT_ADE | A/D error interrupt | 56 | — | Motor control |
| INT_SG1 | Scan group 1 (SG1) end interrupt | 18 | 4 | LPS |
| INT_SG2 | Scan group 2 (SG2) end interrupt | 19 | 5 | LPS |
| INT_SG3 | Scan group 3 (SG3) end interrupt | 20, 32 | 6 | LPS |
| ADC_CONV_END0 | Scan group 4 (SG4) A/D conversion end signal | — | 7 | — |
| **ADCA1** | | | | |
| INT_ADE | A/D error interrupt | 212 | — | — |
| INT_SG1 | Scan group 1 (SG1) end interrupt | 213 | 103 | — |
| INT_SG2 | Scan group 2 (SG2) end interrupt | 214 | 104 | — |
| INT_SG3 | Scan group 3 (SG3) end interrupt | 215 | 105 | — |
| ADC_CONV_END1 | Scan group 4 (SG4) A/D conversion end signal | — | 106 | — |

### 31.1.5 Reset Sources

ADCAn reset sources are listed in the following table. ADCAn is initialized by these reset sources.

**Table 31.9 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| ADCA0 | Reset sources other than transition to DeepSTOP mode (AWORES) |
| ADCA1 | All reset sources (ISORES) |

### 31.1.6 External Input/Output Signals

External input/output signals of ADCAn are listed below.

**Table 31.10 ADCA0 Analog Input Signals**

| Unit Signal Name | Alternative Port Pin Signal | Resolution | T&H | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|---|---|---|
| ANI000 | ADCA0I0 | 12 | √ | √ | √ | √ |
| ANI001 | ADCA0I1 | 12 | √ | √ | √ | √ |
| ANI002 | ADCA0I2 | 12 | √ | √ | √ | √ |
| ANI003 | ADCA0I3 | 12 | √ | √ | √ | √ |
| ANI004 | ADCA0I4 | 12 | √ | √ | √ | √ |
| ANI005 | ADCA0I5 | 12 | √ | √ | √ | √ |
| ANI006 | ADCA0I6 | 12 | — | √ | √ | √ |
| ANI007 | ADCA0I7 | 12 | — | √ | √ | √ |
| ANI008 | ADCA0I8 | 12 | — | √ | √ | √ |
| ANI009 | ADCA0I9 | 12 | — | √ | √ | √ |
| ANI010 | ADCA0I10 | 12 | — | √ | √ | √ |
| ANI011 | ADCA0I11 | 12 | — | √ | √ | √ |
| ANI012 | ADCA0I12 | 12 | — | √ | √ | √ |
| ANI013 | ADCA0I13 | 12 | — | √ | √ | √ |
| ANI014 | ADCA0I14 | 12 | — | √ | √ | √ |
| ANI015 | ADCA0I15 | 12 | — | √ | √ | √ |
| ANI016 | ADCA0I0S | 10 | — | √ | √ | √ |
| ANI017 | ADCA0I1S | 10 | — | √ | √ | √ |
| ANI018 | ADCA0I2S | 10 | — | √ | √ | √ |
| ANI019 | ADCA0I3S | 10 | — | √ | √ | √ |
| ANI020 | ADCA0I4S | 10 | — | √ | √ | √ |
| ANI021 | ADCA0I5S | 10 | — | √ | √ | √ |
| ANI022 | ADCA0I6S | 10 | — | √ | √ | √ |
| ANI023 | ADCA0I7S | 10 | — | √ | √ | √ |
| ANI024 | ADCA0I8S | 10 | — | √ | √ | √ |
| ANI025 | ADCA0I9S | 10 | — | √ | √ | √ |
| ANI026 | ADCA0I10S | 10 | — | √ | √ | √ |
| ANI027 | ADCA0I11S | 10 | — | √ | √ | √ |
| ANI028 | ADCA0I12S | 10 | — | √ | √ | √ |
| ANI029 | ADCA0I13S | 10 | — | √ | √ | √ |
| ANI030 | ADCA0I14S | 10 | — | √ | √ | √ |
| ANI031 | ADCA0I15S | 10 | — | √ | √ | √ |
| ANI032 | ADCA0I16S | 10 | — | √ | √ | √ |
| ANI033 | ADCA0I17S | 10 | — | √ | √ | √ |
| ANI034 | ADCA0I18S | 10 | — | √ | √ | √ |
| ANI035 | ADCA0I19S | 10 | — | √ | √ | √ |

Table 31.11 ADCA0 External Input/Output Signals

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **ADCA0** | | |
| ADCA0TRG0 | External trigger pin (scan group 1)*1 | ADCA0TRG0 |
| ADCA0TRG1 | External trigger pin (scan group 2)*1 | ADCA0TRG1 |
| ADCA0TRG2 | External trigger pin (scan group 3)*1 | ADCA0TRG2 |
| ADCA0SEL0 | External analog multiplexer (MPX) output pin 0 | ADCA0SEL0 |
| ADCA0SEL1 | External analog multiplexer (MPX) output pin 1 | ADCA0SEL1 |
| ADCA0SEL2 | External analog multiplexer (MPX) output pin 2 | ADCA0SEL2 |

Note 1. When the external trigger pin is used, the noise filter for the port needs to be set. For details, see **Section 2.12, Noise Filter & Edge/Level Detector**.

**CAUTION**

**When port P8_6 is used as ADCA0I8S, port P8_6 pin outputs a low-level $\overline{\text{RESETOUT}}$ signal while a reset is asserted and continues to output a low level after the reset is deasserted.**
**For details, see Section 2.11.1.1, P8_6: $\overline{\text{RESETOUT}}$.**

Table 31.12 ADCA1 Analog Input Signals

| Unit Signal Name | Alternative Port Pin Signal | Resolution | T&H | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|---|---|---|
| ANI100 | ADCA1I0 | 12 | — | — | √ | √ |
| ANI101 | ADCA1I1 | 12 | — | — | √ | √ |
| ANI102 | ADCA1I2 | 12 | — | — | √ | √ |
| ANI103 | ADCA1I3 | 12 | — | — | √ | √ |
| ANI104 | ADCA1I4 | 12 | — | — | √ | √ |
| ANI105 | ADCA1I5 | 12 | — | — | √ | √ |
| ANI106 | ADCA1I6 | 12 | — | — | √ | √ |
| ANI107 | ADCA1I7 | 12 | — | — | √ | √ |
| ANI108 | ADCA1I8 | 12 | — | — | — | √ |
| ANI109 | ADCA1I9 | 12 | — | — | — | √ |
| ANI110 | ADCA1I10 | 12 | — | — | — | √ |
| ANI111 | ADCA1I11 | 12 | — | — | — | √ |
| ANI112 | ADCA1I12 | 12 | — | — | — | √ |
| ANI113 | ADCA1I13 | 12 | — | — | — | √ |
| ANI114 | ADCA1I14 | 12 | — | — | — | √ |
| ANI115 | ADCA1I15 | 12 | — | — | — | √ |
| ANI116 | ADCA1I0S | 10 | — | — | √ | √ |
| ANI117 | ADCA1I1S | 10 | — | — | √ | √ |
| ANI118 | ADCA1I2S | 10 | — | — | √ | √ |
| ANI119 | ADCA1I3S | 10 | — | — | √ | √ |
| ANI120 | ADCA1I4S | 10 | — | — | — | √ |
| ANI121 | ADCA1I5S | 10 | — | — | — | √ |
| ANI122 | ADCA1I6S | 10 | — | — | — | √ |
| ANI123 | ADCA1I7S | 10 | — | — | — | √ |

Table 31.13 ADCA1 External Input/Output Signals

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **ADCA1** | | |
| ADCA1TRG0 | External trigger pin (scan group 1)*1 | ADCA1TRG0 |
| ADCA1TRG1 | External trigger pin (scan group 2)*1 | ADCA1TRG1 |
| ADCA1TRG2 | External trigger pin (scan group 3)*1 | ADCA1TRG2 |

Note 1. When the external trigger pin is used, the noise filter for the port needs to be set. For details, see **Section 2.12, Noise Filter & Edge/Level Detector**.

## 31.2 Overview

### 31.2.1 Functional Overview

ADCA has the following features.

- 10-bit/12-bit resolution
- Successive approximation conversion method
- Number of A/D input channels
  A/D conversion is available for a maximum of 36 ADCA0 channels and 24 ADCA1 channels. Additionally, ADCA0 supports the connection of an external analog multiplexer (MPX) to extend the number of analog input channels.
- Internal track and hold (T&H) circuit
  ANI000 to ANI005 (ADCA0I0 to ADCA0I5) of ADCA0 include the track and hold circuit. The track and hold circuit can sample up to 6 channels of analog input simultaneously.
- A/D conversion control by scan groups
  The A/D conversion channel or conversion mode (scan mode) can be set for each scan group.
- Two scan modes
  Multi-cycle scan mode: Specified number of scans are executed.
  Continuous scan mode: Scans are executed repeatedly and continuously.
- Asynchronous/synchronous suspend and resume function
  A processing for a scan group can be interrupted to run the processing for another scan group.
- Start trigger for each scan group
  Software, hardware, and external trigger can start processing of each scan group.
- Scan end interrupt and DMA transfer are supported.
  For each scan group, an interrupt request to INTC can be issued or DMA transfer can be started, each time a processing for the virtual channel indicated by the end virtual channel pointer ends, or a virtual channel ends.
- A/D conversion channel repeat function
  A/D conversion is performed for the same channel two or four times sequentially, and the result is stored in the data register.
- Abundant safety functions
  Abundant safety functions are provided, such as A/D converter diagnosis, diagnosis of the channel multiplexer, diagnosis of open pins, diagnosis of the T&H circuit, upper limit/lower limit check for the A/D converter, overwrite check for data registers, and read and clear function for data registers.
- Shortest A/D conversion time per channel
  1.15 µs (when MPX is not used)
  2.30 µs (when MPX is used)

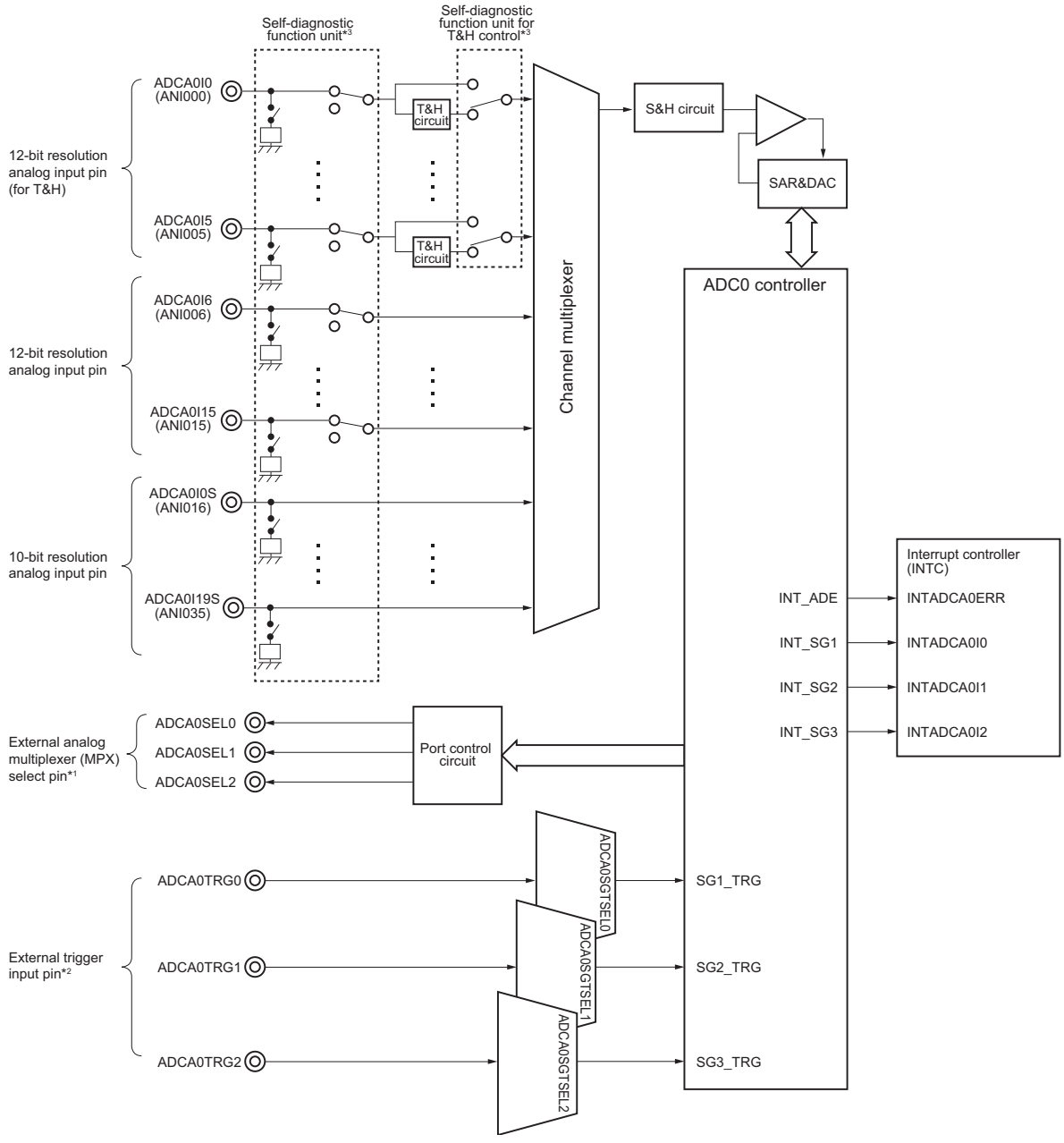
**NOTE**

- Physical channel (ANInm)
  Each A/D input channel of ADCA0 and ADCA1 units is called a physical channel. The physical channel of each unit is represented as ANI0m (m = 0 to 35) for ADCA0 and ANI1m (m = 0 to 23) for ADCA1.
  In RH850/F1K, the alternative port pins for 12-bit resolution A/D input channel and 10-bit resolution A/D input channel are represented as ADCAnIm and ADCAnImS, respectively. In this section, the physical channels and the corresponding alternative port pins are listed together.
- Virtual channel (ADCAnVCRj)
  ADCA0 has a maximum of 50 virtual channels, and ADCA1 has a maximum of 36 virtual channels. The virtual channel specifies the physical channel to be scanned.
  Scans are executed in sequence from the smallest virtual channel number. The scan order can be arbitrarily-specified by using virtual channels. In addition, the scanned result is stored in the data register (ADCAnDRj) corresponding to the virtual channel.
- Scan group (SGx)
  ADCA has three scan groups (SG1, SG2, SG3) and one PWM-Diag group (SG4). A/D conversion is executed in scan group unit. The channel to be scanned can be selected for each group by specifying the scan range, that is, the conversion start virtual channel and the conversion end virtual channel.

## 31.2.2 Block Diagram

The block diagram of ADCA0 is shown in **Figure 31.1**. The block diagram of ADCA1 is shown in **Figure 31.2**.

### (1) Configuration of ADCA0



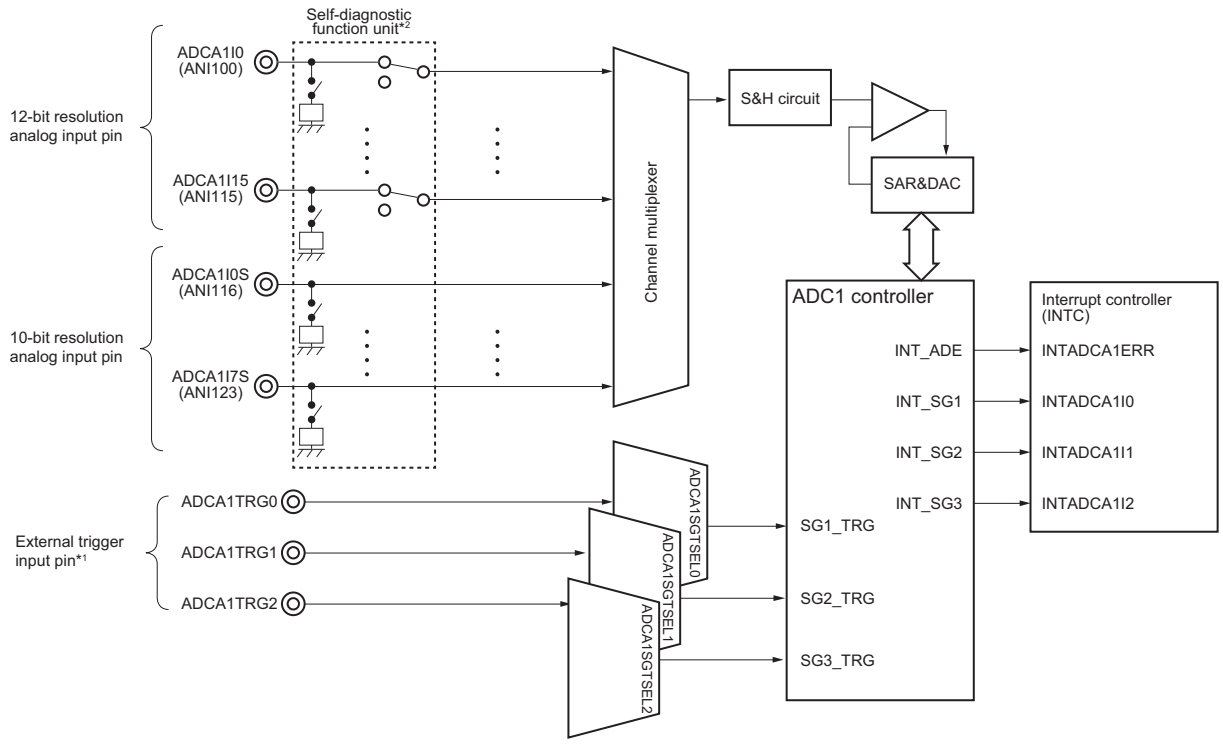

Note 1. For an example of external analog multiplexer connection, see **Figure 31.4, Example of External Analog Multiplexer Connection**.

Note 2. For details, see **Figure 31.3, Internal Connection Diagram of External Trigger Input Pins**.

Note 3. For the self-diagnostic function unit, see **Section 31.5, Self-Diagnostic Function**.

**Figure 31.1 ADCA0 Block Diagram (RH850/F1K with 176 Pins)**

### (2) Configuration of ADCA1



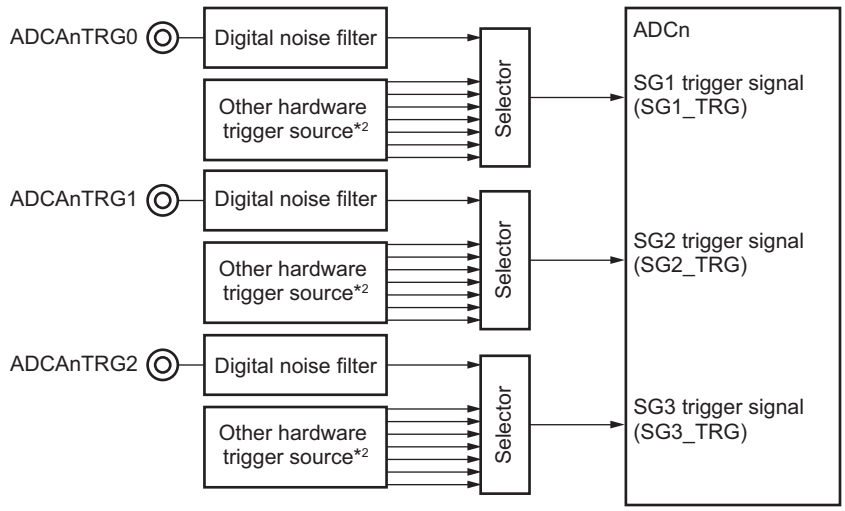

Note 1. For details, see **Figure 31.3, Internal Connection Diagram of External Trigger Input Pins**.
Note 2. For the self-diagnostic function unit, see **Section 31.5, Self-Diagnostic Function**.

**Figure 31.2 ADCA1 Block Diagram (RH850/F1K with 176 Pins)**

### (3) Configuration of external trigger input pins

An external trigger input pin is a hardware trigger source to activate ADCAn.

The configuration of external trigger input pins is shown below.



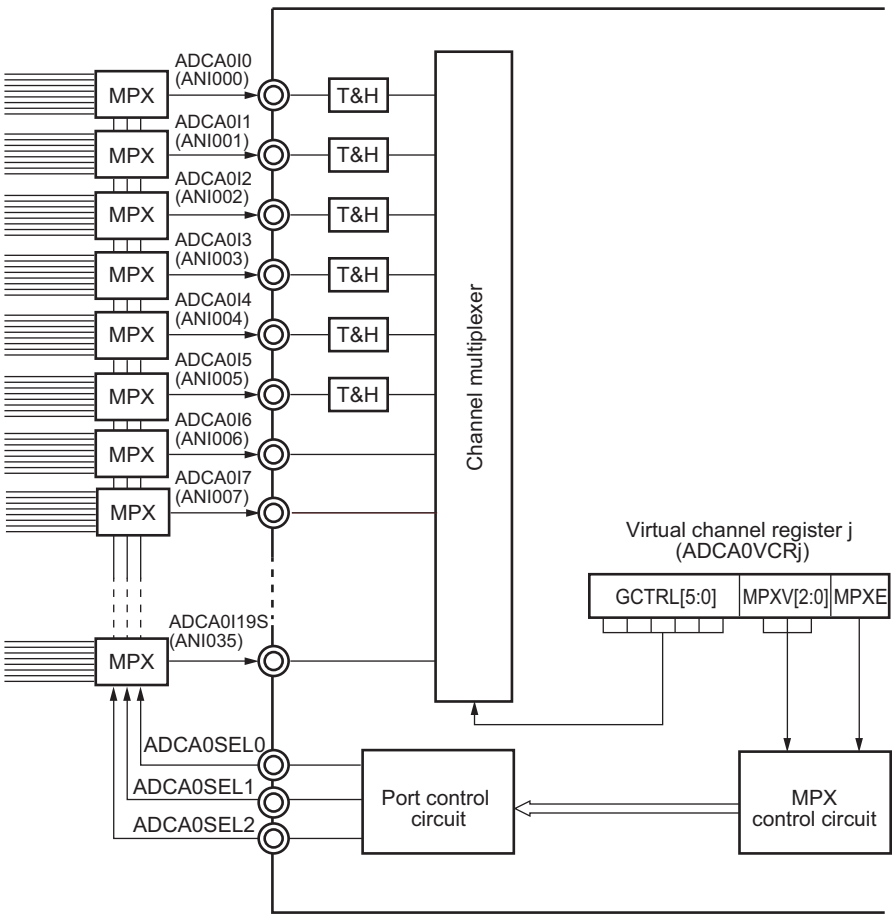

Note 1. For the condition setting to accept a trigger from an external trigger input pin, see **Section 2.12, Noise Filter & Edge/Level Detector**.

Note 2. For the hardware trigger sources, see **Table 31.49, List of A/D Conversion Hardware Triggers**.

**Figure 31.3 Internal Connection Diagram of External Trigger Input Pins**

### (4) Configuration of external analog multiplexer (MPX)

The external analog multiplexer (MPX) can be connected to any input signal pins ADCA0I0 to ADCA0I19S. An example of the external analog multiplexer connection is shown below.

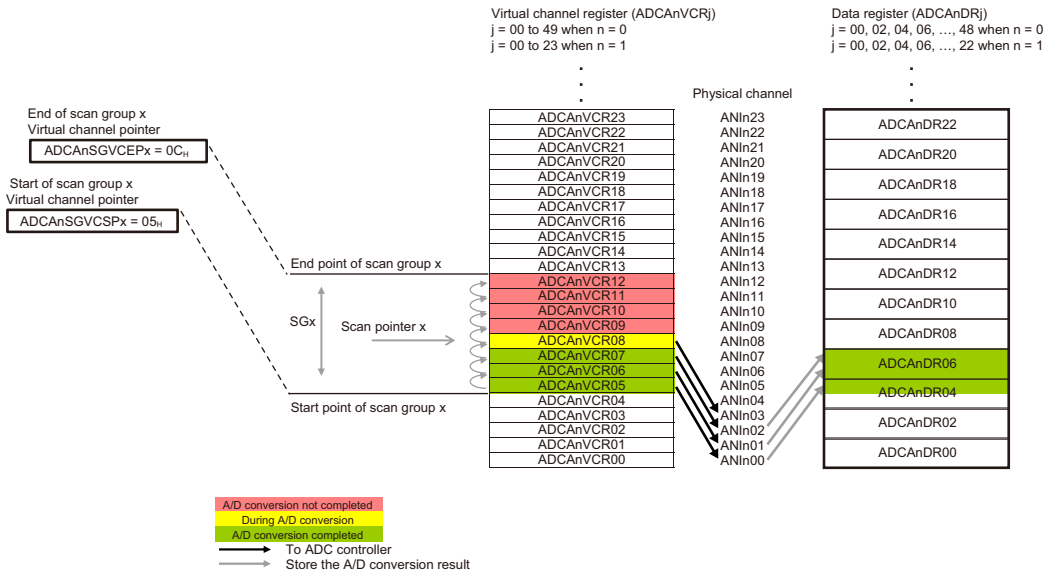



**Note:** The T&H function is disabled when the external analog multiplexer (MPX) is used.

**Figure 31.4 Example of External Analog Multiplexer Connection**

### (5) Virtual channel

The virtual channel specifies the physical address to be scanned.
The virtual channel is controlled by the ADCAnVCRj register.
A usage example of the virtual channel is shown below.




**Example)** When scan group x starts scanning 5th to 12th virtual channels

1. The ADCAnVCRj register specifies the physical channel (ANInm) for which virtual channel j executes the A/D conversion. The A/D conversion pins can be set in any order. (The figure above shows an example when ANIn0 to ANIn7 are selected.)
2. The scan pointer x sequentially reads the physical channels specified in the ADCAnVCRj register from “start point of scan group x” to “end point of scan group x”, and starts the A/D conversion for the read physical address.
3. The scanned result of virtual channel j is stored in data register j.

**Figure 31.5 Usage Example of Virtual Register**

# 31.3 Registers

## 31.3.1 List of Registers

The ADCA registers are listed in the following table.

For details about <ADCAn_base>, see **Section 31.1.2, Register Base Address**.

**Table 31.14 List of Registers (1/2)**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| **ADCA Specific Registers (Virtual Channel)** | | | |
| ADCAn | Virtual Channel Register j | ADCAnVCRj | <ADCAn_base> + j × $4_H$ |
| | PWM-Diag Virtual Channel Register | ADCAnPWDVCR | <ADCAn_base> + $0F4_H$ |
| | Data Register j | ADCAnDRj | <ADCAn_base> + $100_H$ + j × $2_H$ |
| | Data Supplementary Information Register j | ADCAnDIRj | <ADCAn_base> + $200_H$ + j × $4_H$ |
| | PWM-Diag data register | ADCAnPWDTSNDR | <ADCAn_base> + $178_H$ |
| | PWM-Diag Data Supplementary Information Register | ADCAnPWDDIR | <ADCAn_base> + $2F4_H$ |
| **ADCA Specific Registers (Control)** | | | |
| ADCAn | A/D Force Halt Register | ADCAnADHALTR | <ADCAn_base> + $300_H$ |
| | A/D Control Register | ADCAnADCR | <ADCAn_base> + $304_H$ |
| | MPX Current Register | ADCAnMPXCURR | <ADCAn_base> + $30C_H$ |
| | T&H Sampling Start Control Register | ADCAnTHSMPSTCR | <ADCAn_base> + $314_H$ |
| | T&H Control Register | ADCAnTHCR | <ADCAn_base> + $318_H$ |
| | T&H Group A Hold Start Control Register | ADCAnTHAHLDSTCR | <ADCAn_base> + $31C_H$ |
| | T&H Group B Hold Start Control Register | ADCAnTHBHLDSTCR | <ADCAn_base> + $320_H$ |
| | T&H Group A Control Register | ADCAnTHACR | <ADCAn_base> + $324_H$ |
| | T&H Group B Control Register | ADCAnTHBCR | <ADCAn_base> + $328_H$ |
| | T&H Enable Register | ADCAnTHER | <ADCAn_base> + $32C_H$ |
| | T&H Group Select Register | ADCAnTHGSR | <ADCAn_base> + $330_H$ |
| | Sampling Control Register | ADCAnSMPCR | <ADCAn_base> + $380_H$ |
| **ADCA Specific Registers (Safety-related)** | | | |
| ADCAn | Safety Control Register | ADCAnSFTCR | <ADCAn_base> + $334_H$ |
| | Upper Limit/Lower Limit Table Register 0 | ADCAnULLMTBR0 | <ADCAn_base> + $338_H$ |
| | Upper Limit/Lower Limit Table Register 1 | ADCAnULLMTBR1 | <ADCAn_base> + $33C_H$ |
| | Upper Limit/Lower Limit Table Register 2 | ADCAnULLMTBR2 | <ADCAn_base> + $340_H$ |
| | Error Clear Register | ADCAnECR | <ADCAn_base> + $344_H$ |
| | Upper Limit/Lower Limit Error Register | ADCAnULER | <ADCAn_base> + $348_H$ |
| | Overwrite Error Register | ADCAnOWER | <ADCAn_base> + $34C_H$ |
| **Scan Group Specific Registers** | | | |
| ADCAn | Scan Group x Start Control Register | ADCAnSGSTCRx | <ADCAn_base> + x × $40_H$ + $400_H$ |
| | PWM-Diag Scan Group Control Register | ADCAnPWDSGCR | <ADCAn_base> + $508_H$ |
| | Scan Group x Control Register | ADCAnSGCRx | <ADCAn_base> + x × $40_H$ + $408_H$ |
| | Scan Group x Start Virtual Channel Pointer | ADCAnSGVCSPx | <ADCAn_base> + x × $40_H$ + $40C_H$ |
| | Scan Group x End Virtual Channel Pointer | ADCAnSGVCEPx | <ADCAn_base> + x × $40_H$ + $410_H$ |
| | Scan Group x Multicycle Register | ADCAnSGMCYCRx | <ADCAn_base> + x × $40_H$ + $414_H$ |
| | PWM-Diag Scan End Flag Clear Register | ADCAnPWDSGSEFCR | <ADCAn_base> + $518_H$ |
| | Scan Group x Scan End Flag Clear Register | ADCAnSGSEFCRx | <ADCAn_base> + x × $40_H$ + $418_H$ |
| | Scan Group Status Register | ADCAnSGSTR | <ADCAn_base> + $308_H$ |
| **H/W Trigger Specific Register** | | | |
| ADCAn | Scan Group x Start Trigger Control Register | ADCAnSGTSELx | <ADCAn_base> + x × $40_H$ + $41C_H$ |

Table 31.14 List of Registers (2/2)

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| **Self-Diagnosis Specific Registers** | | | |
| ADCAn | Self-Diagnostic Control Register 0 | ADCAnDGCTL0 | <ADCAn_base> + $350_H$ |
| | Self-Diagnostic Control Register 1 | ADCAnDGCTL1 | <ADCAn_base> + $354_H$ |
| | Pull Down Control Register 1 | ADCAnPDCTL1 | <ADCAn_base> + $358_H$ |
| | Pull Down Control Register 2 | ADCAnPDCTL2 | <ADCAn_base> + $35C_H$ |
| **Emulation Specific Register** | | | |
| ADCAn | Emulation Control Register | ADCAnEMU | <ADCAn_base> + $388_H$ |

## 31.3.2 ADCA Specific Registers

This section describes the registers that are equipped in each of ADCA0 and ADCA1.

### 31.3.2.1 ADCAnVCRj — Virtual Channel Register j

This register is used to control the virtual channel.

**Access:** ADCAnVCRj register can be read or written in 32-bit units.
ADCAnVCRjL register can be read or written in 16-bit units.
ADCAnVCRjLL and ADCAnVCRjLH registers can be read or written in 8-bit units.

**Address:** ADCAnVCRj: <ADCAn_base> + j × $4_H$
ADCAnVCRjL: <ADCAn_base> + j × $4_H$
ADCAnVCRjLL: <ADCAn_base> + j × $4_H$
ADCAnVCRjLH: <ADCAn_base> + j × $4_H$ + $1_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MPXE*1 | MPXV[2:0]*1 | | | — | — | CNVCLS*1,*2 | ADIE | ULS[1:0] | | GCTRL[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 31.15 ADCAnVCRj Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 15 | MPXE*1 | MPX Enable<br>0: The use of MPX is prohibited.<br>No wait is inserted before A/D conversion is performed.<br>1: The use of MPX is permitted. The MPXV[2:0] bits are output from ADCAnSEL0 to ADCAnSEL2 when the virtual channel starts, and a wait of one A/D-conversion time is inserted before A/D conversion is performed. |
| 14 to 12 | MPXV[2:0]*1 | These bits are used to set the MPX value to be transferred to an external analog multiplexer. |
| 11, 10 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 9 | CNVCLS*1, *2 | A/D Conversion Type Select for Self-Diagnosis<br>0: A/D conversion of the hold value is performed during a self-diagnosis.<br>1: Normal A/D conversion is performed during a self-diagnosis.<br>When normal A/D conversion is performed during a self-diagnosis and MPX is in use (MPXE is set), however, a wait of one A/D conversion time is inserted before A/D conversion is performed. On the other hand, MPX cannot be used when A/D conversion of the hold value is performed during a self-diagnosis. |
| 8 | ADIE | A/D Conversion End Interrupt Enable<br>0: A scan group x end interrupt (INT_SGx) is not generated when A/D conversion for virtual channel j ends in SGx.<br>1: A scan group x end interrupt (INT_SGx) is generated when A/D conversion for virtual channel j ends in SGx. |
| 7, 6 | ULS[1:0] | Upper Limit/Lower Limit Table Select<br>00: Upper limit and lower limit are not checked.<br>01: Upper limit and lower limit are checked for ADCAnULLMTBR0.<br>10: Upper limit and lower limit are checked for ADCAnULLMTBR1.<br>11: Upper limit and lower limit are checked for ADCAnULLMTBR2. |

**Table 31.15 ADCAnVCRj Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 5 to 0 | GCTRL[5:0] | Physical Channel Select<br>$0_H$ to $23_H$: Corresponding ANInm is selected.<br>$24_H$: Diagnosis channel for A/D converter is selected.<br>Others: Setting prohibit<br>**Note:** These bits shall set the physical channel which is supported. See **Table 31.5, Indices of Products**, **Table 31.10, ADCA0 Analog Input Signals** and **Table 31.12, ADCA1 Analog Input Signals**. |

Note 1. These bits are only supported for ADCA0. For ADCA1, when writing, write the value after reset.

Note 2. This bit is only supported when j = 33 to 35. Otherwise, when writing, write the value after reset.

**CAUTION**

To prevent malfunction, ADCAnVCRj should be set when SGACT of applicable scan groups is 0 (before scan groups are started) and TRGMD of applicable scan groups is 0.

**Table 31.16 Selection of Physical Channels (1/2)**

| GCTRL5 | GCTRL4 | GCTRL3 | GCTRL2 | GCTRL1 | GCTRL0 | Analog Input Pin to be Selected |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | ADCAnI0 (Physical channel ANIn00)*3 |
| 0 | 0 | 0 | 0 | 0 | 1 | ADCAnI1 (Physical channel ANIn01)*3 |
| 0 | 0 | 0 | 0 | 1 | 0 | ADCAnI2 (Physical channel ANIn02)*3 |
| 0 | 0 | 0 | 0 | 1 | 1 | ADCAnI3 (Physical channel ANIn03)*3 |
| 0 | 0 | 0 | 1 | 0 | 0 | ADCAnI4 (Physical channel ANIn04)*3 |
| 0 | 0 | 0 | 1 | 0 | 1 | ADCAnI5 (Physical channel ANIn05)*3 |
| 0 | 0 | 0 | 1 | 1 | 0 | ADCAnI6 (Physical channel ANIn06)*3 |
| 0 | 0 | 0 | 1 | 1 | 1 | ADCAnI7 (Physical channel ANIn07)*3 |
| 0 | 0 | 1 | 0 | 0 | 0 | ADCAnI8 (Physical channel ANIn08)*2, *3 |
| 0 | 0 | 1 | 0 | 0 | 1 | ADCAnI9 (Physical channel ANIn09)*2, *3 |
| 0 | 0 | 1 | 0 | 1 | 0 | ADCAnI10 (Physical channel ANIn10)*2, *3 |
| 0 | 0 | 1 | 0 | 1 | 1 | ADCAnI11 (Physical channel ANIn11)*2, *3 |
| 0 | 0 | 1 | 1 | 0 | 0 | ADCAnI12 (Physical channel ANIn12)*2, *3 |
| 0 | 0 | 1 | 1 | 0 | 1 | ADCAnI13 (Physical channel ANIn13)*2, *3 |
| 0 | 0 | 1 | 1 | 1 | 0 | ADCAnI14 (Physical channel ANIn14)*2, *3 |
| 0 | 0 | 1 | 1 | 1 | 1 | ADCAnI15 (Physical channel ANIn15)*2, *3 |
| 0 | 1 | 0 | 0 | 0 | 0 | ADCAnI0S (Physical channel ANIn16)*3 |
| 0 | 1 | 0 | 0 | 0 | 1 | ADCAnI1S (Physical channel ANIn17)*3 |
| 0 | 1 | 0 | 0 | 1 | 0 | ADCAnI2S (Physical channel ANIn18)*3 |
| 0 | 1 | 0 | 0 | 1 | 1 | ADCAnI3S (Physical channel ANIn19)*3 |
| 0 | 1 | 0 | 1 | 0 | 0 | ADCAnI4S (Physical channel ANIn20)*2, *3 |
| 0 | 1 | 0 | 1 | 0 | 1 | ADCAnI5S (Physical channel ANIn21)*2, *3 |
| 0 | 1 | 0 | 1 | 1 | 0 | ADCAnI6S (Physical channel ANIn22)*2, *3 |
| 0 | 1 | 0 | 1 | 1 | 1 | ADCAnI7S (Physical channel ANIn23)*2, *3 |
| 0 | 1 | 1 | 0 | 0 | 0 | ADCAnI8S (Physical channel ANIn24)*1, *2, *3 |
| 0 | 1 | 1 | 0 | 0 | 1 | ADCAnI9S (Physical channel ANIn25)*1, *2, *3 |
| 0 | 1 | 1 | 0 | 1 | 0 | ADCAnI10S (Physical channel ANIn26)*1, *2, *3 |

**Table 31.16 Selection of Physical Channels (2/2)**

| GCTRL5 | GCTRL4 | GCTRL3 | GCTRL2 | GCTRL1 | GCTRL0 | Analog Input Pin to be Selected |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | 1 | ADCAnI11S (Physical channel ANIn27)*1, *2, *3 |
| 0 | 1 | 1 | 1 | 0 | 0 | ADCAnI12S (Physical channel ANIn28)*1, *2, *3 |
| 0 | 1 | 1 | 1 | 0 | 1 | ADCAnI13S (Physical channel ANIn29)*1, *2, *3 |
| 0 | 1 | 1 | 1 | 1 | 0 | ADCAnI14S (Physical channel ANIn30)*1, *2, *3 |
| 0 | 1 | 1 | 1 | 1 | 1 | ADCAnI15S (Physical channel ANIn31)*1, *2, *3 |
| 1 | 0 | 0 | 0 | 0 | 0 | ADCAnI16S (Physical channel ANIn32)*1, *2, *3 |
| 1 | 0 | 0 | 0 | 0 | 1 | ADCAnI17S (Physical channel ANIn33)*1, *2, *3 |
| 1 | 0 | 0 | 0 | 1 | 0 | ADCAnI18S (Physical channel ANIn34)*1, *2, *3 |
| 1 | 0 | 0 | 0 | 1 | 1 | ADCAnI19S (Physical channel ANIn35)*1, *2, *3 |
| 1 | 0 | 0 | 1 | 0 | 0 | Diagnosis channel for A/D converter |
| Other than above | | | | | | Setting prohibited |

Note 1. This setting only applies to ADCA0. Setting is prohibited in ADCA1 (RH850/F1K with 176 pins).
Note 2. This setting only applies to ADCA0. Setting is prohibited in ADCA1 (RH850/F1K with 144 pins).
Note 3. This setting only applies to ADCA0. Setting is prohibited in ADCA1 (RH850/F1K with 100 pins).

### 31.3.2.2 ADCAnPWDVCR — PWM-Diag Virtual Channel Register

This register is used to indicate virtual channel setting (PWSAnPVCRx_y register setting) of the PWM-Diag (SG4).

**Access:** ADCAnPWDVCR register is a read-only register that can be read in 32-bit units.
ADCAnPWDVCRL register is a read-only register that can be read in 16-bit units.
ADCAnPWDVCRLL register is a read-only register that can be read in 8-bit units.
ADCAnPWDVCRLH register is a read-only register that can be read in 8-bit units.

**Address:** ADCAnPWDVCR: <ADCAn_base> + $0F4_H$
ADCAnPWDVCRL: <ADCAn_base> + $0F4_H$
ADCAnPWDVCRLL: <ADCAn_base> + $0F4_H$
ADCAnPWDVCRLH: <ADCAn_base> + $0F4_H$ + $1_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MPXE*1 | MPXV[2:0]*1 | | | — | — | — | — | ULS[1:0] | | GCTRL[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Table 31.17 ADCAnPWDVCR Register Contents

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. |
| 15 | MPXE*1 | The following setting is made by setting the PWSAnPVCRx_y.PWSAnVRDTy[27] (odd channel) or PWSAnPVCRx_y.PWSAnVRDTx[11] (even channel) bit.<br><br>MPX Enable<br>Set this bit to 1 when an external analog multiplexer is used.<br>0: The use of MPX is prohibited.<br>1: The use of MPX is permitted. The MPXV[2:0] bits are output from ADCAnSEL0 to ADCAnSEL2 when the virtual channel starts, and a wait of one A/D-conversion time is inserted before A/D conversion is performed. |
| 14 to 12 | MPXV[2:0]*1 | These bits are used to set the MPX value to be transferred to an external analog multiplexer by using the PWSAnPVCRx_y.PWSAnVRDTy[26:24] (odd channel) or PWSAnPVCRx_y.PWSAnVRDTx[10:8] (even channel) bit. |
| 11 to 8 | Reserved | When read, the value after reset is returned. |
| 7 to 6 | ULS[1:0] | The following setting is made by setting the PWSAnPVCRx_y.PWSAnVRDTy[23:22] (odd channel) or PWSAnPVCRx_y.PWSAnVRDTx[7:6] (even channel) bit.<br>Upper Limit/Lower Limit Table Select<br>00: Upper limit and lower limit are not checked.<br>01: Upper limit and lower limit are checked for ADCAnULLMTBR0.<br>10: Upper limit and lower limit are checked for ADCAnULLMTBR1.<br>11: Upper limit and lower limit are checked for ADCAnULLMTBR2. |
| 5 to 0 | GCTRL[5:0] | The following setting is made by setting the PWSAnPVCRx_y.PWSAnVRDTy[21:16] (odd channel) or PWSAnPVCRx_y.PWSAnVRDTx[5:0] (even channel) bit.<br><br>Physical Channel Select<br>These bits are used to specify a physical channel to be assigned to virtual channel j.<br>For the selection of the channel, see **Table 31.16, Selection of Physical Channels**. |

Note 1. These bits are only supported for ADCA0. For ADCA1, when read, the value after reset is returned.

### 31.3.2.3 ADCAnDRj — Data Register j

This register is a 32-/16-bit read-only register that stores the A/D conversion results corresponding to ADCAnVCRj and ADCAnVCR(j+1). As the A/D conversion results, the conversion result for ADCAnVCR(j+1) is stored in the upper bits (ADCAnDR(j+1)), and the conversion result for ADCAnVCRj is stored in the lower bits (ADCAnDRj).

**Access:** ADCAnDRj register is a read-only register that can be read in 32-bit units.
ADCAnDRjL and ADCAnDRjH registers are read-only registers that can be read in 16-bit units.

**Address:** ADCAnDRj: <ADCAn_base> + $100_H$ + j × $2_H$
ADCAnDRjL: <ADCAn_base> + $100_H$ + j × $2_H$
ADCAnDRjH: <ADCAn_base> + $100_H$ + j × $2_H$ + $2_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DR(j+1)[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DRj[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 31.18 ADCAnDRj Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | DR(j+1)[15:0] | These bits are used to store the A/D conversion result data.<br>(The A/D conversion result for the channel set in ADCAnVCR(j+1) are transferred.) |
| 15 to 0 | DRj[15:0] | These bits are used to store the A/D conversion result data.<br>(The A/D conversion result for the channel set in ADCAnVCRj are transferred.) |

**CAUTION**

If the number of channels is odd, the higher-order bits (DR(j+1)[15:0]) in the ADCAnDRj register cannot be used. If virtual channels 33, 34, and 35 are used exclusively for self-diagnosis, the lower-order bits (DRj[15:0]) for channel 32 cannot be used.

**NOTES**

1. j = 00, 02, …, 46, 48 (for ADCA0)
   j = 00, 02, …, 32, 34 (for ADCA1 of the RH850/F1K with 176 pins and 144 pins)
2. By controlling ADCAnADCR.CRAC and ADCAnADCR.CTYP, the data format of this register becomes as follows:
   - ADCAnADCR.CTYP = 0 and ADCAnADCR.CRAC = 0 → Right alignment is used.
     →The A/D conversion result for ADCAnVCR(j+1) is transferred to bits 27 to 16, and the A/D conversion result for ADCAnVCRj is transferred to bits 11 to 0.
   - ADCAnADCR.CTYP = 0 and ADCAnADCR.CRAC = 1 → Left alignment is used.
     →The A/D conversion result for ADCAnVCR(j+1) is transferred to bits 31 to 20, and the A/D conversion result for ADCAnVCRj is transferred to bits 15 to 4.
   - ADCAnADCR.CTYP = 1 and ADCAnADCR.CRAC = 0 → Right alignment is used.
     →The A/D conversion result for ADCAnVCR(j+1) is transferred to bits 25 to 16, and the A/D conversion result for ADCAnVCRj is transferred to bits 9 to 0.
   - ADCAnADCR.CTYP = 1 and ADCAnADCR.CRAC = 1 → Left alignment is used.
     →The A/D conversion result for ADCAnVCR(j+1) is transferred to bits 31 to 22, and the A/D conversion result for ADCAnVCRj is transferred to bits 15 to 6.

### 31.3.2.4 ADCAnDIRj — Data Supplementary Information Register j

This register is a 32-bit read-only register that stores the A/D conversion result for ADCAnDRj and information incidental to the A/D converted value.

As the A/D conversion result, the ADCAnDRj value is transferred. As information incidental to the A/D converted value, information about the write flag (WFLG), the MPX value (MPXV[2:0]), and the physical channel (ID[5:0]) is transferred. The data format of the A/D conversion result stored in ADCAnDIRj is the same as that for the ADCAnDRj register.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <ADCAn_base> + $200_H$ + j × $4_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MPXE *1 | MPXV[2:0]*1 | | | — | — | WFLG | — | — | — | ID[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 31.19 ADCAnDIRj Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | MPXE*1 | MPX Enable Flag<br>0: MPX function is not used.<br>1: MPX function is used. |
| 30 to 28 | MPXV[2:0]*1 | These bits are used to store the MPX value.<br>The MPX value to be stored is the MPX value of the most recent conversion result. |
| 27, 26 | Reserved | When read, the value after reset is returned. |
| 25 | WFLG | Write Flag<br>0: ADCAnDRj or ADCAnDIRj is read (cleared when read).<br>1: A/D converted value is stored in ADCAnDRj (set when the value is stored). |
| 24 to 22 | Reserved | When read, the value after reset is returned. |
| 21 to 16 | ID[5:0] | These bits store the physical channel number (GCTRL) corresponding to the conversion result.<br>The physical channel number to be stored is the number corresponding to the most recent conversion result. |
| 15 to 0 | DR[15:0] | These bits are used to store the A/D conversion result. |

Note 1. These bits are only supported by ADCA0.
For ADCA1, when read, the value after reset is returned.

### 31.3.2.5 ADCAnPWDTSNDR — PWM-Diag Data Register

This register is a 32-/16-bit read-only register that stores the A/D conversion results corresponding to the PWM-Diag. As the A/D conversion results, the conversion result for the PWM-Diag (PWDDR) is stored in the upper bits.

**Access:** ADCAnPWDTSNDR register is a read-only register that can be read in 32-bit units.
ADCAnPWDTSNDRH register is a read-only register that can be read in 16-bit units.

**Address:** ADCAnPWDTSNDR: <ADCAn_base> + $178_H$
ADCAnPWDTSNDRH: <ADCAn_base> + $17A_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PWDDR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 31.20 ADCAnPWDTSNDR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | PWDDR[15:0] | These bits are used to store the A/D conversion result data for the PWM-Diag. |
| 15 to 0 | Reserved | When read, the value after reset is returned. |

**NOTE**

The data format of this register is controlled by ADCAnADCR.CRAC and ADCAnADCR.CTYP, as shown below.

- ADCAnADCR.CTYP = 0, ADCAnADCR.CRAC = 0 → Right alignment is used.
  → The A/D conversion result for ADCAnPWDVCR is transferred to bits 27 to 16.
- ADCAnADCR.CTYP = 0, ADCAnADCR.CRAC = 1 → Left alignment is used.
  → The A/D conversion result for ADCAnPWDVCR is transferred to bits 31 to 20.
- ADCAnADCR.CTYP = 1, ADCAnADCR.CRAC = 0 → Right alignment is used.
  → The A/D conversion result for ADCAnPWDVCR is transferred to bits 25 to 16.
- ADCAnADCR.CTYP = 1, ADCAnADCR.CRAC = 1 → Left alignment is used.
  → The A/D conversion result for ADCAnPWDVCR is transferred to bits 31 to 22.

### 31.3.2.6 ADCAnPWDDIR — PWM-Diag Data Supplementary Information Register

This register is a 32-bit read-only register that stores the A/D conversion result when PWM-Diag is used, and information incidental to the A/D converted value.

As the A/D conversion result, the ADCAnPWDSNDR.PWDDR[15:0] value is transferred. As supplementary information to the A/D converted value, the write flag (WFLG), MPX value (MPXV[2:0]), and physical channel (ID[5:0]) are transferred. The data format of the A/D conversion result stored in ADCAnPWDDIR is the same as that for the ADCAnPWDTSNDR register.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** <ADCAn_base> + $2F4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MPXE*1 | MPXV[2:0]*1 | | | — | — | WFLG | — | — | — | ID[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PWDDR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 31.21 ADCAnPWDDIR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | MPXE*1 | MPX Enable Flag<br>0: The MPX function is not used.<br>1: The MPX function is used. |
| 30 to 28 | MPXV[2:0]*1 | These bits are used to store the MPX value.<br>The MPX value to be stored is the MPX value of the most recent conversion result. |
| 27, 26 | Reserved | When read, the value after reset is returned. |
| 25 | WFLG | Write Flag<br>0: ADCAnPWDTSNDR or ADCAnPWDDIR is read (cleared when read).<br>1: The A/D converted value is stored in ADCAnPWDTSNDR (set when the value is stored). |
| 24 to 22 | Reserved | When read, the value after reset is returned. |
| 21 to 16 | ID[5:0] | These bits are used to store the physical channel number (GCTRL) corresponding to the conversion result.<br>The physical channel number to be stored is the physical channel number corresponding to the most recent conversion result. |
| 15 to 0 | PWDDR[15:0] | These bits are used to store the A/D conversion result for PWM-Diag. |

Note 1. These bits are only supported for ADCA0.
For ADCA1, when read, the value after reset is returned.

### 31.3.2.7 ADCAnADHALTR — A/D Force Halt Register

This register is used to halt conversion for all SGs of ADCAn. The read value is always 0.

**Access:** ADCAnADHALTR register is a write-only register that can be written in 32-bit units.
ADCAnADHALTRL register is a write-only register that can be written in 16-bit units.
ADCAnADHALTRLL register is a write-only register that can be written in 8-bit units.

**Address:** ADCAnADHALTR:<ADCAn_base> + $300_H$
ADCAnADHALTRL:<ADCAn_base> + $300_H$
ADCAnADHALTRLL:<ADCAn_base> + $300_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | HALT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 31.22 ADCAnADHALTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | HALT | ADCA Force Halt Trigger<br>All scan groups are halted and initialized, and ADCA becomes idle state.<br>Writing of 0: No effect<br>Writing of 1: Scan groups are halted. |

### 31.3.2.8 ADCAnADCR — A/D Control Register

This register is used for ADCAn common control.

**Access:** ADCAnADCR register can be read or written in 32-bit units.
ADCAnADCRL register can be read or written in 16-bit units.
ADCAnADCRLL register can be read or written in 8-bit units.

**Address:** ADCAnADCR: <ADCAn_base> + $304_H$
ADCAnADCRL: <ADCAn_base> + $304_H$
ADCAnADCRLL: <ADCAn_base> + $304_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | DGON | — | CRAC | CTYP | — | — | SUSMTD[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R | R/W | R/W | R | R | R/W | R/W |

**Table 31.23 ADCAnADCR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 7 | DGON | Self-Diagnostic Voltage Standby Control<br>0: The self-diagnostic voltage circuit is turned off.<br>1: The self-diagnostic voltage circuit is turned on, or the reference voltage is updated. |
| 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 | CRAC | Alignment Control<br>0: The results of conversion to PWDDR and ADCAnDRj are stored right-aligned.<br>1: The results of conversion to PWDDR and ADCAnDRj are stored left-aligned. |
| 4 | CTYP | 12/10 Bit Select Mode<br>0: 12-bit mode<br>1: 10-bit mode |
| 3, 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | SUSMTD [1:0] | Suspend Mode Select<br>These bits are used to select the suspend method when a higher-priority scan group interrupts a lower-priority scan group.<br>00: Synchronous suspend when a higher-priority SG or SVSTOP interrupts.<br>01: Asynchronous suspend when a higher-priority SG (SG2, SG3, SG4) and SVSTOP interrupt SG1, and synchronous suspend when a higher-priority SG (SG3, SG4) and SVSTOP interrupt SG2, or when a higher-priority SG (SG4) and SVSTOP interrupt SG3.<br>10: Asynchronous suspend when a higher-priority SG or SVSTOP interrupts.<br>11: Setting prohibited |

**CAUTION**

To prevent malfunction, ADCAnADCR should be set when SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.

**NOTE**

- Synchronous suspend:
  If a request from a higher-priority SG occurs while a lower-priority SG is being processed, the A/D conversion for the higher-priority SG is performed after the on-going A/D conversion of a channel is completed. After processing for the higher-priority SG is completed, the suspended A/D channel processing for the lower-priority SG is resumed.
- Asynchronous suspend:
  If a request from a higher-priority SG occurs while a lower-priority SG is being processed, the on-going channel processing is suspended, and then the A/D conversion for the higher-priority SG is performed. After processing for the higher-priority SG is completed, the suspended A/D channel conversion for the lower-priority SG is resumed.

For details, see **Figure 31.21, Example of Synchronous Suspend and Resume Operation** and **Figure 31.22, Example of Asynchronous Suspend and Resume Operation**.

### 31.3.2.9 ADCAnMPXCURR — MPX Current Register

This register is used to store the MPX value for an external analog multiplexer.

**Access:** ADCAnMPXCURR register is a read-only register that can be read in 32-bit units.
ADCAnMPXCURRL register is a read-only register that can be read in 16-bit units.
ADCAnMPXCURRLL register is a read-only register that can be read in 8-bit units.

**Address:** ADCAnMPXCURR: <ADCAn_base> + $30C_H$
ADCAnMPXCURRL: <ADCAn_base> + $30C_H$
ADCAnMPXCURRLL: <ADCAn_base> + $30C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | MPXCUR[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 31.24 ADCAnMPXCURR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. |
| 2 to 0 | MPXCUR[2:0] | These bits are used to store the current MPX value.<br>If conversion of a virtual channel starts after setting ADCAnVCRj.MPXE to 1, the setting of ADCAnVCRj.MPXV[2:0] is stored.<br>If conversion of a virtual channel starts after setting ADCAnPWDVCR.MPXE to 1, the setting of ADCAnPWDVCR.MPXV[2:0] is stored. |

**NOTE**

In RH850/F1K, only ADCA0 supports this function.

### 31.3.2.10 ADCAnTHSMPSTCR — T&H Sampling Start Control Register

This register is used to control the start of sampling for all T&Hk (k = 0 to 5). The bits are always read as 0.

**Access:** ADCAnTHSMPSTCR register is a write-only register that can be written in 32-bit units.
ADCAnTHSMPSTCRL register is a write-only register that can be written in 16-bit units.
ADCAnTHSMPSTCRLL register is a write-only register that can be written in 8-bit units.

**Address:** ADCAnTHSMPSTCR: <ADCAn_base> + $314_H$
ADCAnTHSMPSTCRL: <ADCAn_base> + $314_H$
ADCAnTHSMPSTCRLL: <ADCAn_base> + $314_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SMPST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 31.25 ADCAnTHSMPSTCR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | SMPST | T&H Sampling Start Control Trigger<br>0: No effect<br>1: Sampling for all T&H is started.<br><br>The conditions to place the T&H circuit in the sampling state are as follows:<br>• Condition to start sampling while T&H is stopped:<br>1 being written to ADCAnTHSMPSTCR.SMPST while ADCAnTHER.THkE = 1 (k = 0 to 5).<br>• Condition to start continuous sampling in automatic sampling:<br>A/D conversion of the hold value for T&Hk being completed while ADCAnTHER.THkE = 1 (k = 0 to 5) and ADCAnTHCR.ASMPMSK = 1. |

**NOTE**

In RH850/F1K, only ADCA0 supports this function.

### 31.3.2.11 ADCAnTHCR — T&H Control Register

This register controls the sampling transition after A/D conversion of the hold value for T&H is completed.

Automatic start of sampling on the T&H circuit after A/D conversion of the hold value for T&H is completed shortens the time required for the generation of succeeding hold completion triggers.

**Access:** ADCAnTHCR register can be read or written in 32-bit units.
ADCAnTHCRL register can be read or written in 16-bit units.
ADCAnTHCRLL register can be read or written in 8-bit units.

**Address:** ADCAnTHCR: <ADCAn_base> + $318_H$
ADCAnTHCRL: <ADCAn_base> + $318_H$
ADCAnTHCRLL: <ADCAn_base> + $318_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ASMPMSK |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 31.26 ADCAnTHCR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | ASMPMSK | Automatic Sampling Mask Control<br>0: Automatic sampling is not performed.<br>1: Automatic sampling is performed. |

**CAUTION**

To prevent malfunction, ADCAnTHCR should be set when SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.

**NOTE**

In RH850/F1K, only ADCA0 supports this function.

### 31.3.2.12 ADCAnTHAHLDSTCR — T&H Group A Hold Start Control Register

This register is used to control the start of the hold for T&H group A. The bits are always read as 0.

**Access:** ADCAnTHAHLDSTCR register is a write-only register that can be written in 32-bit units.
ADCAnTHAHLDSTCRL register is a write-only register that can be written in 16-bit units.
ADCAnTHAHLDSTCRLL register is a write-only register that can be written in 8-bit units.

**Address:** ADCAnTHAHLDSTCR: <ADCAn_base> + $31C_H$
ADCAnTHAHLDSTCRL: <ADCAn_base> + $31C_H$
ADCAnTHAHLDSTCRLL: <ADCAn_base> + $31C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | HLDST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 31.27 ADCAnTHAHLDSTCR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | HLDST | T&H Group A Hold Start Control Trigger<br>0: No effect<br>1: Hold for T&H group A is started.<br><br>The condition to place T&H group A in the hold state is as follows:<br>• 1 being written to ADCAnTHAHLDSTCR.HLDST while ADCAnTHER.THkE = 1 (k = 0 to 5) and ADCAnTHGSR.THkGS = 0 (k = 0 to 5). |

**NOTE**

In RH850/F1K, only ADCA0 supports this function.

### 31.3.2.13 ADCAnTHBHLDSTCR — T&H Group B Hold Start Control Register

This register is used to control the start of the hold for T&H group B. The bits are always read as 0.

**Access:** ADCAnTHBHLDSTCR register is a write-only register that can be written in 32-bit units.
ADCAnTHBHLDSTCRL register is a write-only register that can be written in 16-bit units.
ADCAnTHBHLDSTCRLL register is a write-only register that can be written in 8-bit units.

**Address:** ADCAnTHBHLDSTCR: <ADCAn_base> + $320_H$
ADCAnTHBHLDSTCRL: <ADCAn_base> + $320_H$
ADCAnTHBHLDSTCRLL: <ADCAn_base> + $320_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | HLDST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 31.28 ADCAnTHBHLDSTCR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | HLDST | T&H Group B Hold Start Control Trigger<br>0: No effect<br>1: Hold for T&H group B is started.<br><br>The condition to place T&H group B in the hold state is as follows:<br>• 1 being written to ADCAnTHBHLDSTCR.HLDST while ADCAnTHER.THkE = 1 (k = 0 to 5) and ADCAnTHGSR.THkGS = 1 (k = 0 to 5). |

**NOTE**

In RH850/F1K, only ADCA0 supports this function.

### 31.3.2.14 ADCAnTHACR — T&H Group A Control Register

This register is used to control T&H group A.

**Access:** ADCAnTHACR register can be read or written in 32-bit units.
ADCAnTHACRL register can be read or written in 16-bit units.
ADCAnTHACRLL register can be read or written in 8-bit units.

**Address:** ADCAnTHACR: <ADCAn_base> + $324_H$
ADCAnTHACRL: <ADCAn_base> + $324_H$
ADCAnTHACRLL: <ADCAn_base> + $324_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | HLDCTE | HLDTE | — | — | SGS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R/W | R/W |

**Table 31.29 ADCAnTHACR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 | HLDCTE | T&H Group A Hold Completion Trigger Enable<br>This bit is used when self-diagnosis of the T&H circuit is to proceed.<br>0: Self-diagnosis does not proceed.<br>1: Self-diagnosis proceeds.<br><br>**Note:** The SGx_TRG (x = 1 to 3) trigger is selected for the trigger input of the scan group that is not selected in SGS[1:0] of ADCAnTHACR and SGS[1:0] of ADCAnTHBCR. |
| 4 | HLDTE | T&H Group A Hold Trigger Enable<br>0: The SGx (x = 1 to 3) trigger selected in SGS[1:0] is selected for the hold start trigger of T&H group A.<br>1: The SGx (x = 1 to 3) trigger selected in SGS[1:0] is not selected for the hold start trigger of T&H group A.<br>**Note:** ADCAnTHAHLDSTCR.HLDST becomes a hold start trigger regardless of the ADCAnTHACR.HLDTE setting.<br>Set this bit to 0 when self-diagnosis of the T&H circuit is to proceed. |
| 3, 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | SGS[1:0] | T&H Group A Scan Group Select<br>00: No scan group is selected for T&H group A.<br>01: SG1 is selected for T&H group A.<br>10: SG2 is selected for T&H group A.<br>11: SG3 is selected for T&H group A.<br>**Note:**<br>1. If ADCAnTHACR.SGS[1:0] is set to $0_H$, T&H does not operate. When you enable T&Hk in ADCAnTHER.THkE, make sure that a scan group is specified in SGS[1:0].<br>2. Selecting the same scan group as T&H group B is prohibited. |

**CAUTION**

To prevent malfunction, ADCAnTHACR should be set when SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.

**NOTE**

In RH850/F1K, only ADCA0 supports this function.

### 31.3.2.15 ADCAnTHBCR — T&H Group B Control Register

This register is used to control T&H group B.

**Access:** ADCAnTHBCR register can be read or written in 32-bit units.
ADCAnTHBCRL register can be read or written in 16-bit units.
ADCAnTHBCRLL register can be read or written in 8-bit units.

**Address:** ADCAnTHBCR: <ADCAn_base> + $328_H$
ADCAnTHBCRL: <ADCAn_base> + $328_H$
ADCAnTHBCRLL: <ADCAn_base> + $328_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | HLDCTE | HLDTE | — | — | SGS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R/W | R/W |

**Table 31.30 ADCAnTHBCR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 | HLDCTE | T&H Group B Hold Completion Trigger Enable<br>This bit is used when self-diagnosis of the T&H circuit is to proceed.<br>0: Self-diagnosis does not proceed.<br>1: Self-diagnosis proceeds.<br><br>**Note:** The SGx_TRG (x = 1 to 3) trigger is selected for the trigger input of the scan group that is not selected in SGS[1:0] of ADCAnTHACR and SGS[1:0] of ADCAnTHBCR. |
| 4 | HLDTE | T&H Group B Hold Trigger Enable<br>0: The SGx (x = 1 to 3) trigger selected in SGS[1:0] is selected for the hold start trigger of T&H group B.<br>1: The SGx (x = 1 to 3) trigger selected in SGS[1:0] is not selected for the hold start trigger of T&H group B.<br><br>**Note:** ADCAnTHBHLDSTCR.HLDST becomes a hold start trigger regardless of the ADCAnTHBCR.HLDTE setting.<br>Set this bit to 0 when self-diagnosis of the T&H circuit is to proceed. |
| 3, 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

Table 31.30 ADCAnTHBCR Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 1 to 0 | SGS[1:0] | T&H Group B Scan Group Select<br>00: No scan group is selected for T&H group B.<br>01: SG1 is selected for T&H group B.<br>10: SG2 is selected for T&H group B.<br>11: SG3 is selected for T&H group B.<br><br>**Note:**<br>1. If ADCAnTHBCR.SGS[1:0] is set to $0_H$, T&H does not operate. When you enable T&Hk in ADCAnTHER.THkE, make sure that a scan group is specified in SGS[1:0].<br>2. Selecting the same scan group as T&H group A is prohibited. |

**CAUTION**

To prevent malfunction, ADCAnTHBCR should be set when SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.

**NOTE**

In RH850/F1K, only ADCA0 supports this function.

### 31.3.2.16 ADCAnTHER — T&H Enable Register

This register controls enabling and disabling of each T&H.

**Access:** ADCAnTHER register can be read or written in 32-bit units.
ADCAnTHERL register can be read or written in 16-bit units.
ADCAnTHERLL register can be read or written in 8-bit units.

**Address:** ADCAnTHER: <ADCAn_base> + $32C_H$
ADCAnTHERL: <ADCAn_base> + $32C_H$
ADCAnTHERLL: <ADCAn_base> + $32C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | TH5E | TH4E | TH3E | TH2E | TH1E | TH0E |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 31.31 ADCAnTHER Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 | TH5E | T&H5 Enable<br>0: T&H5 is disabled.<br>1: T&H5 is enabled.<br>**Note:** If TH5E is set to 0, T&H5 is always stopped. |
| 4 | TH4E | T&H4 Enable<br>0: T&H4 is disabled.<br>1: T&H4 is enabled.<br>**Note:** If TH4E is set to 0, T&H4 is always stopped. |
| 3 | TH3E | T&H3 Enable<br>0: T&H3 is disabled.<br>1: T&H3 is enabled.<br>**Note:** If TH3E is set to 0, T&H3 is always stopped. |
| 2 | TH2E | T&H2 Enable<br>0: T&H2 is disabled.<br>1: T&H2 is enabled<br>**Note:** If TH2E is set to 0, T&H2 is always stopped. |
| 1 | TH1E | T&H1 Enable<br>0: T&H1 is disabled.<br>1: T&H1 is enabled.<br>**Note:** If TH1E is set to 0, T&H1 is always stopped. |
| 0 | TH0E | T&H0 Enable<br>0: T&H0 is disabled.<br>1: T&H0 is enabled<br>**Note:** If TH0E is set to 0, T&H0 is always stopped. |

**CAUTION**

To prevent malfunction, ADCAnTHER should be set when SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.

**NOTE**

In RH850/F1K, only ADCA0 supports this function.

### 31.3.2.17 ADCAnTHGSR — T&H Group Select Register

This register is used to select a T&H group for each T&H.

**Access:** ADCAnTHGSR register can be read or written in 32-bit units.
ADCAnTHGSRL register can be read or written in 16-bit units.
ADCAnTHGSRLL register can be read or written in 8-bit units.

**Address:** ADCAnTHGSR: <ADCAn_base> + $330_H$
ADCAnTHGSRL: <ADCAn_base> + $330_H$
ADCAnTHGSRLL: <ADCAn_base> + $330_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | TH5GS | TH4GS | TH3GS | TH2GS | TH1GS | TH0GS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 31.32 ADCAnTHGSR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 | TH5GS | T&H5 Group Select<br>0: T&H5 is selected to group A.<br>1: T&H5 is selected to group B. |
| 4 | TH4GS | T&H4 Group Select<br>0: T&H4 is selected to group A.<br>1: T&H4 is selected to group B. |
| 3 | TH3GS | T&H3 Group Select<br>0: T&H3 is selected to group A.<br>1: T&H3 is selected to group B. |
| 2 | TH2GS | T&H2 Group Select<br>0: T&H2 is selected to group A.<br>1: T&H2 is selected to group B. |
| 1 | TH1GS | T&H1 Group Select<br>0: T&H1 is selected to group A.<br>1: T&H1 is selected to group B. |
| 0 | TH0GS | T&H0 Group Select<br>0: T&H0 is selected to group A.<br>1: T&H0 is selected to group B. |

**CAUTION**

- Do not set T&H0 to T&H2 to the same group as T&H3 to T&H5.
  Example
  - Group A: 0ch, 1ch, 2ch
    Group B: 3ch, 4ch, 5ch → Setting allowed
  - Group A: 0ch
    Group B: 1ch, 2ch → Setting allowed
  - Group A: 0ch, 1ch, 3ch
    Group B: 2ch, 4ch → Setting prohibited
- To prevent malfunction, ADCAnTHGSR should be set when SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.

**NOTE**

In RH850/F1K, only ADCA0 supports this function.

### 31.3.2.18 ADCAnSMPCR — Sampling Control Register

This register controls sampling.

ADCTLnSMPCR controls the sampling time for SG4 (PWM-Diag) and SG1 to SG3.

**Access:** ADCAnSMPCR register can be read or written in 32-bit units.
ADCAnSMPCRL register can be read or written in 16-bit units.
ADCAnSMPCRLL register can be read or written in 8-bit units.

**Address:** ADCAnSMPCR: <ADCAn_base> + $380_H$
ADCAnSMPCRL: <ADCAn_base> + $380_H$
ADCAnSMPCRLL: <ADCAn_base> + $380_H$

**Value after reset:** 0000 $0018_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | SMPT[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 31.33 ADCAnSMPCR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 7 to 0 | SMPT[7:0] | These bits are used to set the sampling time (the number of cycles).<br>$12_H$: 18 cycles (ADCLK = 8 MHz to 32 MHz)<br>$18_H$: 24 cycles (ADCLK = 8 MHz to 40 MHz)<br>Settings other than above are prohibited. |

**CAUTION**

- To prevent malfunction, ADCATLnSMPCR should be set when SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.
- When SMPT is changed, the A/D conversion wait time is also changed when MPX is used by virtual channel register j (ADCAnVCRj) or PWM-Diag virtual channel register (ADCAnPWDVCR).

### 31.3.2.19 ADCAnSFTCR — Safety Control Register

This is a register for safety control.

**Access:** ADCAnSFTCR register can be read or written in 32-bit units.
ADCAnSFTCRL register can be read or written in 16-bit units.
ADCAnSFTCRLL register can be read or written in 8-bit units.

**Address:** ADCAnSFTCR: <ADCAn_base> + $334_H$
ADCAnSFTCRL: <ADCAn_base> + $334_H$
ADCAnSFTCRLL: <ADCAn_base> + $334_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | RDCLRE | ULEIE | OWEIE | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R | R |

**Table 31.34 ADCAnSFTCR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 4 | RDCLRE | Read & Clear Enable<br>When the A/D conversion result is read, this bit selects whether the A/D conversion result is cleared by hardware.<br>0: ADCAnPWDTSNDR/ADCAnDRj and ADCAnPWDDIR/ADCAnDIRj are not cleared when ADCAnPWDTSNDR/ADCAnDRj or ADCAnPWDDIR/ADCAnDIRj is read.<br>1: ADCAnPWDTSNDR/ADCAnDRj and ADCAnPWDDIR/ADCAnDIRj are cleared when ADCAnPWDTSNDR/ADCAnDRj or ADCAnPWDDIR/ADCAnDIRj is read.<br><br>WFLG of ADCAnDIRj is cleared regardless of the RDCLRE setting when ADCAnDRj or ADCAnDIRj is read. |
| 3 | ULEIE | A/D Error Interrupt (INT_ADE) Enable on Upper/Lower Limit Error Detection<br>0: Disabled<br>1: Enabled |
| 2 | OWEIE | A/D Error Interrupt (INT_ADE) Enable on Overwrite Error Detection<br>0: Disabled<br>1: Enabled |
| 1, 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

**CAUTION**

To prevent malfunction, ADCAnSFTCR should be set when SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.

### 31.3.2.20 ADCAnULLMTBR0 to 2 — Upper Limit/Lower Limit Table Registers 0 to 2

These registers are used to set the threshold for detection of an upper limit or lower limit error in the A/D converted value. Any of ADCAnULLMTBR0 to ADCAnULLMTBR2 is specified by setting ADCAnPWDVCR.ULS[1:0] and ADCAnVCRj.ULS[1:0] and compared with ADCAnPWDTSNDR and ADCAnDRj.

**Access:** ADCAnULLMTBR0 to 2 register can be read or written in 32-bit units.
ADCAnULLMTBR0L to 2L and ADCAnULLMTBR0H to 2H registers can be read or written in 16-bit units.

**Address:** ADCAnULLMTBR0: <ADCAn_base> + $338_H$
ADCAnULLMTBR1: <ADCAn_base> + $33C_H$
ADCAnULLMTBR2: <ADCAn_base> + $340_H$

ADCAnULLMTBR0L: <ADCAn_base> + $338_H$
ADCAnULLMTBR1L: <ADCAn_base> + $33C_H$
ADCAnULLMTBR2L: <ADCAn_base> + $340_H$

ADCAnULLMTBR0H: <ADCAn_base> + $338_H$ + $2_H$
ADCAnULLMTBR1H: <ADCAn_base> + $33C_H$+ $2_H$
ADCAnULLMTBR2H: <ADCAn_base> + $340_H$ + $2_H$

**Value after reset:** $FFF0\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ULMTB[11:0] | | | | | | | | | | | | — | — | — | — |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LLMTB[11:0] | | | | | | | | | | | | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |

**Table 31.35 ADCAnULLMTBR0 to 2 Registers Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | ULMTB[11:0] | Upper Limit Table<br>Specify the threshold for detection of an upper limit error in the A/D converted value. The upper limit error (ADCAnULER.UE) is set when the following condition is met:<br>ULMTB[11:0] < A/D converted value |
| 19 to 16 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 15 to 4 | LLMTB[11:0] | Lower Limit Table<br>Specify the threshold for detection of a lower limit error in the A/D converted value. The lower limit error (ADCAnULER.LE) is set when the following condition is met:<br>LLMTB[11:0] > A/D converted value |
| 3 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

**CAUTION**

- When A/D conversion is executed in 10-bit mode (ADCAnADCR.CTYP = 1), ULMTB[1:0] and LLMTB[1:0] should be set to $11_B$ and $00_B$, respectively.
- To prevent malfunction, ADCAnULLMTBR0 to 2 should be set when SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.
- The upper-limit table (ULMTB[11:0]) must be greater than the lower-limit table (LLMTB[11:0]).

### 31.3.2.21 ADCAnECR — Error Clear Register

This register is used to control clearing of an error. The read value is always 0.

**Access:** ADCAnECR register is a write-only register that can be written in 32-bit units.
ADCAnECRL register is a write-only register that can be written in 16-bit units.
ADCAnECRLL register is a write-only register that can be written in 8-bit units.

**Address:** ADCAnECR: <ADCAn_base> + $344_H$
ADCAnECRL: <ADCAn_base> + $344_H$
ADCAnECRLL: <ADCAn_base> + $344_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | ULEC | OWEC | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | W | W | R | R |

**Table 31.36 ADCAnECR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When writing, write the value after reset. |
| 3 | ULEC | Upper Limit Error Flag (ADCAnULER.UE) / Lower Limit Error Flag (ADCAnULER.LE) Clear, Upper/Lower Limit Error Capture (ADCAnULER.ULECAP[5:0]), Scan Group Bit (ULSG[1:0]) When Upper/Lower Limit Error Occurs, MPX Usage Bit (MPXE), and the MPX Value Storing Bit (MPXV[2:0]) Clear When Upper/Lower Limit Error Occurs<br>0: No effect.<br>1: Clears the flag. |
| 2 | OWEC | Overwrite Error Flag (ADCAnOWER.OWE) and Overwrite Error Capture (ADCAnOWER.OWECAP[5:0]) Clear<br>0: No effect.<br>1: Clears the flag. |
| 1, 0 | Reserved | When writing, write the value after reset. |

### 31.3.2.22 ADCAnULER — Upper Limit/Lower Limit Error Register

This register is a read-only register that indicates information regarding the upper limit/lower limit errors.

**Access:** ADCAnULER register is a read-only register that can be read in 32-bit units.
ADCAnULERL register is a read-only register that can be read in 16-bit units.

ADCAnULERLH register is a read-only register that can be read in 8-bit units.
ADCAnULERLL register is a read-only register that can be read in 8-bit units.

**Address:** ADCAnULER: <ADCAn_base> + $348_H$
ADCAnULERL: <ADCAn_base> + $348_H$

ADCAnULERLL: <ADCAn_base> + $348_H$
ADCAnULERLH: <ADCAn_base> + $348_H$ + $1_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UE | LE | ULSG[1:0] | | MPXE | MPXV[2:0] | | | — | — | ULECAP[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 31.37 ADCAnULER Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. |
| 15 | UE | Upper Limit Error Flag<br>0: An upper limit error is not detected.<br>1: An upper limit error is detected.<br>Setting condition:<br>The A/D converted value exceeds the upper limit threshold specified by the upper limit/lower limit table registers 0 to 2 (ADCAnULLMTBR0 to 2).<br>If a subsequent upper limit error is detected in A/D conversion while this bit is set to 1, the ADCAnULER register is not updated.<br>Clearing condition:<br>1 is written to ADCAnECR.ULEC. |
| 14 | LE | Lower Limit Error Flag<br>0: A lower limit error is not detected.<br>1: A lower limit error is detected.<br>Setting condition:<br>The A/D converted value is lower than the lower limit threshold specified by the upper limit/lower limit table registers 0 to 2 (ADCAnULLMTBR0 to 2).<br>If a subsequent lower limit error is detected in A/D conversion while this bit is set to 1, the ADCAnULER register is not updated.<br>Clearing condition:<br>1 is written to ADCAnECR.ULEC. |
| 13, 12 | ULSG[1:0] | Scan Group where an Upper Limit/Lower Limit Error Occurs<br>00: No upper limit/lower limit error occurred.<br>01: The scan group where an upper limit/lower limit error occurred is SG1 to SG3.<br>10: The scan group where an upper limit/lower limit error occurred is PWM-Diag.<br>Capture condition:<br>When the A/D converted value exceeds the range of the specified upper- or lower-limit table while UE = 0 and LE = 0.<br>Clearing condition:<br>When 1 is written to ADCAnECR.ULEC. |

**Table 31.37 ADCAnULER Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 | MPXE*1 | MPX Usage<br>0: The MPX function was not used when an upper limit/lower limit error occurred.<br>1: The MPX function was used when an upper limit/lower limit error occurred.<br>Capture condition:<br>When the A/D converted value exceeds the range of the specified upper- or lower-limit table while UE = 0 and LE = 0<br>Clearing condition:<br>When 1 is written to ADCAnECR.ULEC. |
| 10 to 8 | MPXV[2:0]*1 | The value of MPX is stored when the errors of the upper and lower limit occurred<br>Capture condition:<br>When the A/D converted value exceeds the range of the specified upper- or lower-limit table while UE = 0 and LE = 0.<br>Clearing condition:<br>When 1 is written to ADCAnECR.ULEC. |
| 7 | Reserved | When read, an undefined value is read. |
| 6 | Reserved | When read, the value after reset is returned. |
| 5 to 0 | ULECAP[5:0] | Upper Limit/Lower Limit Error Capture<br>The physical channel is captured when an upper limit/lower limit error occurred.<br>Capturing condition:<br>When the A/D converted value exceeds the range of the specified upper- or lower-limit table while UE = 0 and LE = 0.<br>Clearing condition:<br>1 is written to ADCAnECR.ULEC. |

Note 1. These bits are only supported for ADCA0.
For ADCA1, when read, the value after reset is returned.

**NOTE**

ADCAnULER is updated when the A/D converted value is set in ADCAnDRj or ADCAnPWDTSNDR.

### 31.3.2.23 ADCAnOWER — Overwrite Error Register

This register is a 32/16/8-bit read-only register that indicates an overwrite error. The target for overwrite errors is SG1 to SG3, and not PWM-Diag.

**Access:** ADCAnOWER register is a read-only register that can be read in 32-bit units.
ADCAnOWERL register is a read-only register that can be read in 16-bit units.
ADCAnOWERLL register is a read-only register that can be read in 8-bit units.

**Address:** ADCAnOWER: <ADCAn_base> + $34C_H$
ADCAnOWERL: <ADCAn_base> + $34C_H$
ADCAnOWERLL: <ADCAn_base> + $34C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | OWE | — | OWECAP[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 31.38 ADCAnOWER Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is returned. |
| 7 | OWE | Overwrite Error Flag<br>0: An overwrite error is not detected.<br>1: An overwrite error is detected.<br>Setting condition:<br>ADCAnDIRj.WFLG = 1, and the A/D converted value is written to ADCAnDRj. If a subsequent overwrite error is detected in A/D conversion while this bit is set to 1, the ADCAnOWER register is not updated.<br>Clearing condition:<br>1 is written to ADCAnECR.OWEC. |
| 6 | Reserved | When read, the value after reset is returned. |
| 5 to 0 | OWECAP[5:0] | Overwrite Error Capture<br>The virtual channel number is captured when an overwrite error occurs.<br>Capturing condition:<br>OWE = 0 and ADCAnDIRj.WFLG = 1, and the A/D converted value is written to ADCAnDRj<br>Clearing condition:<br>1 is written to ADCAnECR.OWEC. |

**NOTE**

ADCAnOWER is updated when the A/D converted value is set in ADCAnDRj.

## 31.3.3 Scan Group (SG) Specific Registers

This section describes the registers provided for each scan group.

### 31.3.3.1 ADCAnSGSTCRx — Scan Group x Start Control Register

This register is used to control the start of scan group x. The read value is always 0.

**Access:** ADCAnSGSTCRx is a write-only register that can be written in 32-bit units.
ADCAnSGSTCRxL is a write-only register that can be written in 16-bit units.
ADCAnSGSTCRxLL is a write-only register that can be written in 8-bit units.

**Address:** ADCAnSGSTCRx: <ADCAn_base> + $400_H$ + x × $40_H$
ADCAnSGSTCRxL: <ADCAn_base> + $400_H$ + x × $40_H$
ADCAnSGSTCRxLL: <ADCAn_base> + $400_H$ + x × $40_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SGST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 31.39 ADCAnSGSTCRx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | SGST | Scan Group Start Trigger<br>Writing 1 to SGST while ADCAnSGSTR.SGACT[3:1] = 0 starts the target SGx. |

### 31.3.3.2 ADCAnSGCRx — Scan Group x Control Register

This register controls scan group x.

**Access:** ADCAnSGCRx register can be read or written in 32-bit units.
ADCAnSGCRxL register can be read or written in 16-bit units.
ADCAnSGCRxLL register can be read or written in 8-bit units.

**Address:** ADCAnSGCRx: <ADCAn_base> + x × $40_H$ + $408_H$
ADCAnSGCRxL: <ADCAn_base> + x × $40_H$ + $408_H$
ADCAnSGCRxLL: <ADCAn_base> + x × $40_H$ + $408_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | SCANMD | ADIE | SCT[1:0] | | — | TRGMD |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R | R/W |

**Table 31.40 ADCAnSGCRx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 | SCANMD | Scan Mode<br>0: Multicycle scan mode<br>1: Continuous scan mode<br>Write 0 to this bit for SG2 and SG3. |
| 4 | ADIE | Scan End Interrupt Enable<br>0: INT_SGx is not output when the scan for SGx ends.<br>1: INT_SGx is output when the scan for SGx ends. |
| 3, 2 | SCT[1:0] | Channel Repeat Times Select<br>00: The selected number of channel repeat times is one.<br>01: The selected number of channel repeat times is two.<br>10: The selected number of channel repeat times is four.<br>11: Setting prohibited |
| 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | TRGMD | Trigger Mode<br>0: Trigger input to SGx_TRG is disabled (Hardware trigger disabled).<br>1: SGx_TRG start trigger or hold complete trigger A/B is selected for the trigger input to SGx.<br>**NOTE**<br>The software trigger is valid regardless of the TRGMD bit setting. |

**CAUTION**

To prevent malfunction, ADCAnSGCRx should be set (except clearing TRGMD upon completion of A/D conversion) when SGACT of all scan groups is 0 (before the scan group is started) and TRGMD of all scan groups is 0.

### 31.3.3.3 ADCAnPWDSGCR — PWM-Diag Scan Group Control Register

This register is used to control PWM-Diag.

**Access:** ADCAnPWDSGCR register can be read or written in 32-bit units.
ADCAnPWDSGCRL register can be read or written in 16-bit units.
ADCAnPWDSGCRLL register can be read or written in 8-bit units.

**Address:** ADCAnPWDSGCR: <ADCAn_base> + $508_H$
ADCAnPWDSGCRL: <ADCAn_base> + $508_H$
ADCAnPWDSGCRLL: <ADCAn_base> + $508_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PWDTRGMD |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 31.41 ADCAnPWDSGCR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | PWDTRGMD | PWM-Diag Trigger Mode Select<br>0: PWSA_ADTRG trigger input is disabled.<br>1: PWSA_ADTRG is selected for the trigger input to the PWM-Diag scan group. |

**CAUTION**

To prevent malfunction, ADCAnPWDSGCR should be set when SGACT of the PWM-Diag scan group (SG4) is 0 (before the scan group is started).

### 31.3.3.4 ADCAnSGVCSPx — Scan Group x Start Virtual Channel Pointer

This register specifies the start pointer of a virtual channel.

**Access:** ADCAnSGVCSPx register can be read or written in 32-bit units.
ADCAnSGVCSPxL register can be read or written in 16-bit units.
ADCAnSGVCSPxLL register can be read or written in 8-bit units.

**Address:** ADCAnSGVCSPx: <ADCAn_base> + x × $40_H$ + $40C_H$
ADCAnSGVCSPxL: <ADCAn_base> + x × $40_H$ + $40C_H$
ADCAnSGVCSPxLL: <ADCAn_base> + x × $40_H$ + $40C_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | VCSP[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 31.42 ADCAnSGVCSPx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 to 0 | VCSP[5:0] | Start Virtual Channel Pointer<br>These bits are used to specify the virtual channel from which the SGx scan is to be started. |

**CAUTION**

- ADCAnSGVCSPx must be equal to or smaller than ADCAnSGVCEPx.
- When writing to the channel pointers, be sure to write in the following order: ADCAnSGVCSPx → ADCAnSGVCEPx. When SGx is started, the A/D conversion for the virtual channels within the range specified in ADCAnSGVCSPx and ADCAnSGVCEPx is executed.
- Though ADCAnSGVCSPx can be written during the A/D conversion, the register is updated at the time when ADCAnSGVCEPx is written. The new setting is applied when SGx is started next time.
- When the hardware trigger is used, writing to this register during operation is prohibited.

### 31.3.3.5 ADCAnSGVCEPx — Scan Group x End Virtual Channel Pointer

This register specifies the end pointer of a virtual channel.

**Access:** ADCAnSGVCEPx register can be read or written in 32-bit units.
ADCAnSGVCEPxL register can be read or written in 16-bit units.
ADCAnSGVCEPxLL register can be read or written in 8-bit units.

**Address:** ADCAnSGVCEPx: <ADCAn_base> + x × $40_H$ + $410_H$
ADCAnSGVCEPxL: <ADCAn_base> + x × $40_H$ + $410_H$
ADCAnSGVCEPxLL: <ADCAn_base> + x × $40_H$ + $410_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | VCEP[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 31.43 ADCAnSGVCEPx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 5 to 0 | VCEP[5:0] | End Virtual Channel Pointer<br>These bits are used to specify the virtual channel at which the SGx scan is to be ended. |

**CAUTION**

- ADCAnSGVCSPx must be equal to or smaller than ADCAnSGVCEPx.
- When SGx is started, processing for the virtual channels within the range specified in ADCAnSGVCSPx and ADCAnSGVCEPx is executed.
  ADCAnSGVCEPx can be rewritten even when SGx is being processed. The new setting is applied when SGx is started next time.

### 31.3.3.6 ADCAnSGMCYCRx — Scan Group x Multicycle Register

This register is a 32/16/8-bit read/write register that indicates the number of scan times in multicycle scan mode.

**Access:** ADCAnSGMCYCRx register can be read or written in 32-bit units.
ADCAnSGMCYCRxL register can be read or written in 16-bit units.
ADCAnSGMCYCRxLL register can be read or written in 8-bit units.

**Address:** ADCAnSGMCYCRx: <ADCAn_base> + x × $40_H$ + $414_H$
ADCAnSGMCYCRxL: <ADCAn_base> + x × $40_H$ + $414_H$
ADCAnSGMCYCRxLL: <ADCAn_base> + x × $40_H$ + $414_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MCYC[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 31.44 ADCAnSGMCYCRx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1, 0 | MCYC[1:0] | Multicycle Number Specification<br>These bits are used to specify the number of scan times in multicycle scan mode.<br>$00_B$: Number of scans = 1<br>$01_B$: Number of scans = 2<br>$10_B$: Setting prohibited<br>$11_B$: Number of scans = 4 |

**CAUTION**

- To prevent malfunction, ADCAnSGMCYCRx should be set when SGACT of scan group x is 0 (before the scan group is started) and TRGMD is 0.
- When SGx is started, the scan for the virtual channels within the range specified in ADCAnSGVCSPx and ADCAnSGVCEPx is repeatedly executed as many times as specified in ADCAnSGMCYCRx.

### 31.3.3.7 ADCAnPWDSGSEFCR — PWM-Diag Scan End Flag Clear Register

This register is used to control the clearing of PWM-Diag scan end flag (SEF). The bits are always read as 0.

**Access:** ADCAnPWDSGSEFCR register is a write-only register that can be written in 32-bit units.
ADCAnPWDSGSEFCRL register is a write-only register that can be written in 16-bit units.
ADCAnPWDSGSEFCRLL register is a write-only register that can be written in 8-bit units.

**Address:** ADCAnPWDSGSEFCR: <ADCAn_base> + $518_H$
ADCAnPWDSGSEFCRL: <ADCAn_base> + $518_H$
ADCAnPWDSGSEFCRLL: <ADCAn_base> + $518_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PWDSEFC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 31.45 ADCAnPWDSGSEFCR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | PWDSEFC | PWM-Diag Scan End Flag Clear Trigger<br>0: No effect.<br>1: Clears the PWM-Diag scan end flag (ADCAnSGSTR.SEF[4]). |

### 31.3.3.8 ADCAnSGSEFCRx — Scan Group x Scan End Flag Clear Register

This register is a write-only register that clears the scan end flag (ADCAnSGSTR.SEFx). The read value is always 0.

**Access:** ADCAnSGSEFCRx register is a write-only register that can be written in 32-bit units.
ADCAnSGSEFCRxL register is a write-only register that can be written in 16-bit units.
ADCAnSGSEFCRxLL register is a write-only register that can be written in 8-bit units.

**Address:** ADCAnSGSEFCRx: <ADCAn_base> + x × $40_H$ + $418_H$
ADCAnSGSEFCRxL: <ADCAn_base> + x × $40_H$ + $418_H$
ADCAnSGSEFCRxLL: <ADCAn_base> + x × $40_H$ + $418_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SEFC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 31.46 ADCAnSGSEFCRx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | SEFC | Scan End Flag Clear Trigger<br>0: No effect.<br>1: Clears the target SG scan end flag (ADCAnSGSTR.SEF[3:1]). |

### 31.3.3.9 ADCAnSGSTR — Scan Group Status Register

This register indicates the state of T&H, SVSTOP, scan group x, and PWM-Diag scan group. The SHACT and SGACT bits are cleared when HALT is executed.

**Access:** ADCAnSGSTR register is a read-only register that can be read in 32-bit units.
ADCAnSGSTRL register is a read-only register that can be read in 16-bit units.

**Address:** ADCAnSGSTR: <ADCAn_base> + $308_H$
ADCAnSGSTRL: <ADCAn_base> + $308_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | SHACT | SGACT[5:1] | | | | | — | — | — | — | SEF[4:1] | | | | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 31.47 ADCAnSGSTR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 15 | Reserved | When read, the value after reset is returned. |
| 14 | SHACT | T&H Status Flag<br>0: T&H is stopped.<br>1: T&H conversion or sampling is in progress. |
| 13 | SGACT[5] | SVSTOP Status Flag<br>0: SVSTOP is canceled.<br>1: SVSTOP is accepted. |
| 12 | SGACT[4] | PWM-Diag Scan Group (SG4) Status Flag<br>0: A/D conversion for PWM-Diag (SG4) is completed.<br>1: A/D conversion for PWM-Diag (SG4) is in processing or suspension. |
| 11 | SGACT[3] | Scan Group 3 (SG3) Status Flag<br>0: A/D conversion for SG3 is completed.<br>1: A/D conversion for SG3 is in processing or suspension. |
| 10 | SGACT[2] | Scan Group 2 (SG2) Status Flag<br>0: A/D conversion for SG2 is completed.<br>1: A/D conversion for SG2 is in processing or suspension. |
| 9 | SGACT[1] | Scan Group 1 (SG1) Status Flag<br>0: A/D conversion for SG1 is completed.<br>1: A/D conversion for SG1 is in processing or suspension. |
| 8 to 5 | Reserved | When read, the value after reset is returned. |
| 4 | SEF[4] | PWM-Diag Scan End Flag<br>Indicates the status of the scan result data.<br>0: The flag is cleared when any of the following operations is performed:<br>• ADCAnPWDTSNDR for PWM-Diag is read.<br>• ADCAnPWDDIR for PWM-Diag is read.<br>• ADCAnPWDSGSEFCR.PWDSEFC is written as 1.<br>1: The A/D conversion result is written to ADCAnPWDTSNDR for PWM-Diag. |

Table 31.47 ADCAnSGSTR Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 3 | SEF[3] | SG3 Scan End Flag<br>Indicates the status of the scan result data.<br>0: The flag is cleared when any of the following operations is performed:<br>• ADCAnDRj for the virtual channel which ADCAnSGVCEP3 indicates is read.<br>• ADCAnDIRj for the virtual channel which ADCAnSGVCEP3 indicates is read.<br>• ADCAnSGSEFCRx.SEFC is written as 1.<br>1: The A/D conversion result is written to ADCAnDRj for the virtual channel which ADCAnSGVCEP3 indicates. |
| 2 | SEF[2] | SG2 Scan End Flag<br>Indicates the status of the scan result data.<br>0: The flag is cleared when any of the following operations is performed:<br>• ADCAnDRj for the virtual channel which ADCAnSGVCEP2 indicates is read.<br>• ADCAnDIRj for the virtual channel which ADCAnSGVCEP2 indicates is read.<br>• ADCAnSGSEFCRx.SEFC is written as 1.<br>1: The A/D conversion result is written to ADCAnDRj for the virtual channel which ADCAnSGVCEP2 indicates. |
| 1 | SEF[1] | SG1 Scan End Flag<br>Indicates the status of the scan result data.<br>0: The flag is cleared when any of the following operations is performed:<br>• ADCAnDRj for the virtual channel which ADCAnSGVCEP1 indicates is read.<br>• ADCAnDIRj for the virtual channel which ADCAnSGVCEP1 indicates is read.<br>• ADCAnSGSEFCRx.SEFC is written as 1.<br>1: The A/D conversion result is written to ADCAnDRj for the virtual channel which ADCAnSGVCEP1 indicates. |
| 0 | Reserved | When read, the value after reset is returned. |

## 31.3.4 Hardware Trigger Specific Register

### 31.3.4.1 ADCAnSGTSELx — Scan Group x Start Trigger Control Register x

This register is used to select the A/D conversion trigger (hardware trigger) for SGx.

**Access:** ADCAnSGTSELx register can be read or written in 32-bit units.
ADCAnSGTSELxL register can be read or written in 16-bit units.

**Address:** ADCAnSGTSELx: <ADCAn_base> + x × $40_H$ + $41C_H$
ADCAnSGTSELxL: <ADCAn_base> + x × $40_H$ + $41C_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | TxSEL8 *1 | TxSEL7 *1 | TxSEL6 *1 | TxSEL5 *1 | TxSEL4 *1 | TxSEL3 *1 | TxSEL2 *1 | TxSEL1 *1 | TxSEL0 *1 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. ADCA1 supports only TxSEL0 to TxSEL3. When writing to the other bits, write the value after reset.

**Table 31.48 ADCAnSGTSELx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 8 to 0 | TxSELp (p = 0 to 8) | A/D Conversion Trigger (Hardware Trigger) Select<br>0: Hardware trigger is disabled.<br>1: Hardware trigger is enabled.<br>**CAUTION**<br>When setting TxSELp to 1, set only one of the bits to 1. |

The list below shows the hardware triggers to be selected.

**Table 31.49 List of A/D Conversion Hardware Triggers (1/2)**

| Unit | Control Register/Bit | | Trigger Input Signal | |
|---|---|---|---|---|
| | Register Name | Bit Name | Symbol | Connection Destination Unit |
| ADCA0 | ADCA0SGTSEL1 | T1SEL0 | ADCA0TRG0 | External trigger pin |
| | | T1SEL1 | INTTAUJ0I3 | TAUJ0 |
| | | T1SEL2 | INTTAUD0I7 | TAUD0 |
| | | T1SEL3 | INTTAUD0I15 | TAUD0 |
| | | T1SEL4 | SEQADTRG | LPS |
| | | T1SEL5 | INTENCA0I1 | ENCA0 |
| | | T1SEL6 | TAPATADOUT0 | Motor control (TAPA0) |
| | | T1SEL7 | TAPATADOUT1 | Motor control (TAPA0) |
| | | T1SEL8 | ADOPA0ADCATTIN00 | Motor control (PIC0) |

Table 31.49 List of A/D Conversion Hardware Triggers (2/2)

| Unit | Control Register/Bit | | Trigger Input Signal | |
|---|---|---|---|---|
| | Register Name | Bit Name | Symbol | Connection Destination Unit |
| ADCA0 | ADCA0SGTSEL2 | T2SEL0 | ADCA0TRG1 | External trigger pin |
| | | T2SEL1 | INTTAUJ0I3 | TAUJ0 |
| | | T2SEL2 | INTTAUD0I7 | TAUD0 |
| | | T2SEL3 | INTTAUD0I15 | TAUD0 |
| | | T2SEL4 | SEQADTRG | LPS |
| | | T2SEL5 | INTENCA0I1 | ENCA0 |
| | | T2SEL6 | TAPATADOUT0 | Motor control (TAPA0) |
| | | T2SEL7 | TAPATADOUT1 | Motor control (TAPA0) |
| | | T2SEL8 | ADOPA1ADCATTIN00 | Motor control (PIC0) |
| | ADCA0SGTSEL3 | T3SEL0 | ADCA0TRG2 | External trigger pin |
| | | T3SEL1 | INTTAUJ0I3 | TAUJ0 |
| | | T3SEL2 | INTTAUD0I7 | TAUD0 |
| | | T3SEL3 | INTTAUD0I15 | TAUD0 |
| | | T3SEL4 | SEQADTRG | LPS |
| | | T3SEL5 | INTENCA0I1 | ENCA0 |
| | | T3SEL6 | TAPATADOUT0 | Motor control (TAPA0) |
| | | T3SEL7 | TAPATADOUT1 | Motor control (TAPA0) |
| | | T3SEL8 | ADOPA2ADCATTIN00 | Motor control (PIC0) |
| ADCA1 | ADCA1SGTSEL1 | T1SEL0 | ADCA1TRG0 | External trigger pin |
| | | T1SEL1 | INTTAUJ1I3 | TAUJ1 |
| | | T1SEL2 | INTTAUB0I7 | TAUB0 |
| | | T1SEL3 | INTTAUB0I15 | TAUB0 |
| | ADCA1SGTSEL2 | T2SEL0 | ADCA1TRG1 | External trigger pin |
| | | T2SEL1 | INTTAUJ1I3 | TAUJ1 |
| | | T2SEL2 | INTTAUB0I7 | TAUB0 |
| | | T2SEL3 | INTTAUB0I15 | TAUB0 |
| | ADCA1SGTSEL3 | T3SEL0 | ADCA1TRG2 | External trigger pin |
| | | T3SEL1 | INTTAUJ1I3 | TAUJ1 |
| | | T3SEL2 | INTTAUB0I7 | TAUB0 |
| | | T3SEL3 | INTTAUB0I15 | TAUB0 |

**CAUTIONS**

1. When enabling the LPS trigger factor (SEQADTRG), select and enable only one of ADCA0SGTSEL1.T1SEL4, ADCA0SGTSEL2.T2SEL4, and ADCA0SGTSEL3.T3SEL4.
2. To prevent malfunction, ADCAnSGTSELx should be set when SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.

## 31.3.5 Self-Diagnosis Specific Registers

### 31.3.5.1 ADCAnDGCTL0 — Self-Diagnosis Control Register 0

This register controls the self-diagnostic voltage level.

**Access:** ADCAnDGCTL0 register can be read or written in 32-bit units.
ADCAnDGCTL0L register can be read or written in 16-bit units.
ADCAnDGCTL0LL register can be read or written in 8-bit units.

**Address:** ADCAnDGCTL0: <ADCAn_base> + $350_H$
ADCAnDGCTL0L: <ADCAn_base> + $350_H$
ADCAnDGCTL0LL: <ADCAn_base> + $350_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | PSEL2 | PSEL1 | PSEL0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 31.50 ADCAnDGCTL0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2 to 0 | PSEL[2:0] | Self-Diagnostic Voltage Level Select |

| ADCAnDGCTL0 | | | Output Signal | | | |
|---|---|---|---|---|---|---|
| PSEL2 | PSEL1 | PSEL0 | ADDIAGOUT | DIAGOUT2 | DIAGOUT1 | DIAGOUT0 |
| 0 | 0 | 0 | Hi-z | Hi-z | Hi-z | Hi-z |
| 0 | 0 | 1 | AnVSS | $2/3AnV_{REF}$ | $1/2AnV_{REF}$ | $1/3AnV_{REF}$ |
| 0 | 1 | 0 | $1/3AnV_{REF}$ | $1/3AnV_{REF}$ | $2/3AnV_{REF}$ | $1/2AnV_{REF}$ |
| 0 | 1 | 1 | $1/2AnV_{REF}$ | $1/2AnV_{REF}$ | $1/3AnV_{REF}$ | $2/3AnV_{REF}$ |
| 1 | 0 | 0 | $2/3AnV_{REF}$ | Hi-z | Hi-z | Hi-z |
| 1 | 0 | 1 | $AnV_{REF}$ | $1/3AnV_{REF}$ | $1/3AnV_{REF}$ | $1/3AnV_{REF}$ |
| 1 | 1 | 0 | $AnV_{REF}$ | $1/2AnV_{REF}$ | $1/2AnV_{REF}$ | $1/2AnV_{REF}$ |
| 1 | 1 | 1 | $AnV_{REF}$ | $2/3AnV_{REF}$ | $2/3AnV_{REF}$ | $2/3AnV_{REF}$ |

### 31.3.5.2 ADCAnDGCTL1 — Self-Diagnosis Control Register 1

This register controls the self-diagnostic channel.

**Access:** ADCAnDGCTL1 register can be read or written in 32-bit units.
ADCAnDGCTL1L register can be read or written in 16-bit units.

**Address:** ADCAnDGCTL1: <ADCAn_base> + $354_H$
ADCAnDGCTL1L: <ADCAn_base> + $354_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CDG[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 31.51 ADCAnDGCTL1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 15, 12, 9, 6, 3, 0 | CDG [15, 12, 9, 6, 3, 0] | Self-Diagnostic Channel Select<br>0: ANInm is selected.<br>1: DIAGOUT0 is selected. |
| 13, 10, 7, 4, 1 | CDG [13, 10, 7, 4, 1] | Self-Diagnostic Channel Select<br>0: ANInm is selected.<br>1: DIAGOUT1 is selected. |
| 14, 11, 8, 5, 2 | CDG [14, 11, 8, 5, 2] | Self-Diagnostic Channel Select<br>0: ANInm is selected.<br>1: DIAGOUT2 is selected. |

**CAUTION**

To prevent malfunction, ADCAnDGCTL1 should be set when SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.

### 31.3.5.3 ADCAnPDCTL1 — Pull Down Control Register 1

This register specifies the channel to which the pull down resistor is connected.

For details, see **Section 31.5.3, Diagnosis of Open Pins**.

**Access:** ADCAnPDCTL1 register can be read or written in 32-bit units.
ADCAnPDCTL1L register can be read or written in 16-bit units.

**Address:** ADCAnPDCTL1: <ADCAn_base> + $358_H$
ADCAnPDCTL1L: <ADCAn_base> + $358_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PDNA[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 31.52 ADCAnPDCTL1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 15 to 0 | PDNA[15:0] | Pull Down Enable Control<br>These bits set whether an on-chip pull-down resistor is to be connected to the corresponding physical channel (ANIn[00:15]).<br>0: An on-chip pull-down resistor is not connected.<br>1: An on-chip pull-down resistor is connected. |

**CAUTION**

To prevent malfunction, ADCAnPDCTL1 should be set when SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.

**NOTE**

For on-chip pull-down resistor values, see the Electrical Characteristics section in the Data Sheet document.

### 31.3.5.4 ADCAnPDCTL2 — Pull Down Control Register 2

This register specifies the channel to which the pull down resistor is connected.

For details, see **Section 31.5.3, Diagnosis of Open Pins**.

**Access:** ADCAnPDCTL2 register can be read or written in 32-bit units.
ADCAnPDCTL2H and ADCAnPDCTL2L registers can be read or written in 16-bit units.
ADCAnPDCTL2HL, ADCAnPDCTL2LH, and ADCAnPDCTL2LL registers can be read or written in 8-bit units.

**Address:** ADCAnPDCTL2: <ADCAn_base> + $35C_H$
ADCAnPDCTL2L: <ADCAn_base> + $35C_H$
ADCAnPDCTL2H: <ADCAn_base> + $35C_H$ + $2_H$
ADCAnPDCTL2HL: <ADCAn_base> + $35C_H$ +$2_H$
ADCAnPDCTL2LL: <ADCAn_base> + $35C_H$
ADCAnPDCTL2LH: <ADCAn_base> + $35C_H$ + $1_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | PDNB[19:16]*1 | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PDNB[15:0]*1 | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. ADCA1 supports only PDNB[7:0]. When writing to the other bits, write the value after reset.

**Table 31.53 ADCAnPDCTL2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 19 to 0 | PDNB[19:0] | Pull Down Enable Control<br>These bits set whether the on-chip pull-down resistor is to be connected with the corresponding physical channel (ANIn[16:35]).<br>0: The on-chip pull-down resistor is not connected.<br>1: The on-chip pull-down resistor is connected. |

**CAUTION**

To prevent malfunction, ADCAnPDCTL2 should be set when SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.

**NOTE**

For on-chip pull-down resistor values, see the Electrical Characteristics section in the Data Sheet document.

## 31.3.6 Emulation Specific Register

### 31.3.6.1 ADCAnEMU — Emulation Control Register

This register controls the SVSTOP disable signal.

**Access:** This register can be read or written in 8-bit units.

**Address:** <ADCAn_base> + $388_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SVSDIS | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R |

**Table 31.54 ADCAnEMU Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | SVSDIS | SVSTOP Disable<br>0: SVSTOP is enabled<br>1: SVSTOP is disabled<br>For the A/D conversion when SVSTOP is enabled, see **Section 31.4.10.3, SVSTOP Operation**. |
| 6 to 0 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

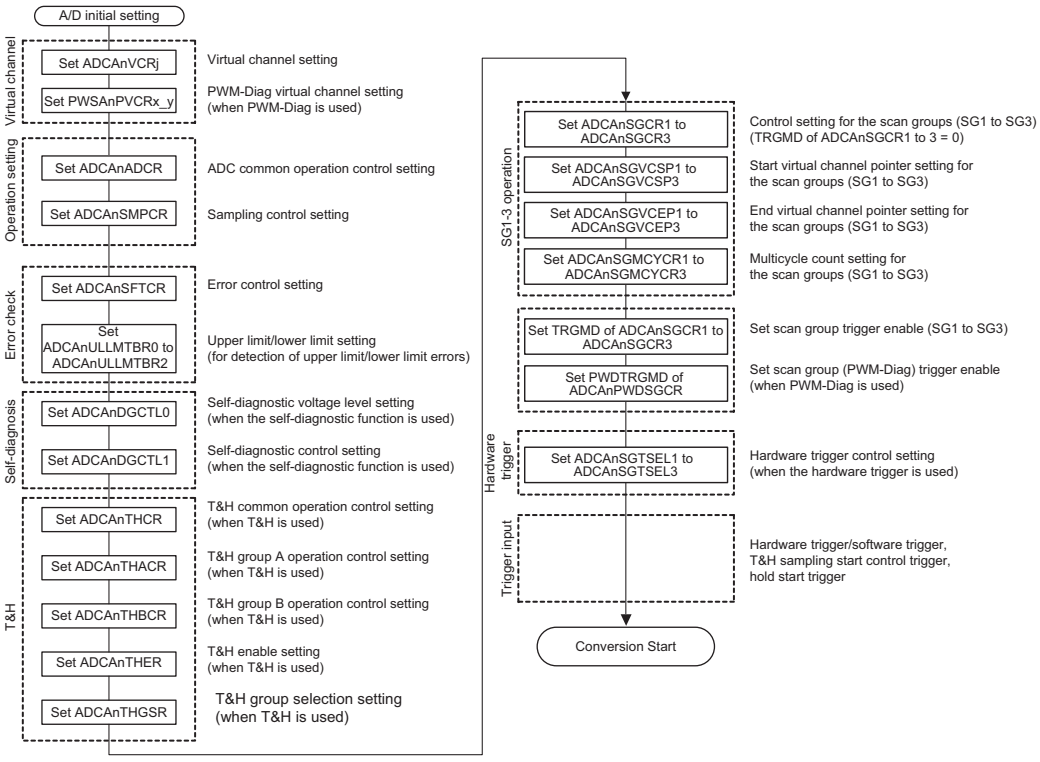
**CAUTION**

To prevent malfunction, SVSDIS should be set when SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.

# 31.4 Operation

## 31.4.1 Initial Setting

**Figure 31.6** shows an initial setting example of the A/D conversion. For trigger input, see **Figure 31.7** in the next section. For interrupt request signals, see **Section 31.4.12, Scan End Interrupt Request**.




**Note:** Set these registers as necessary.

**Figure 31.6 Flowchart for Initial Setting**

### 31.4.2 Trigger Input

The following figure shows the flowchart for trigger input.

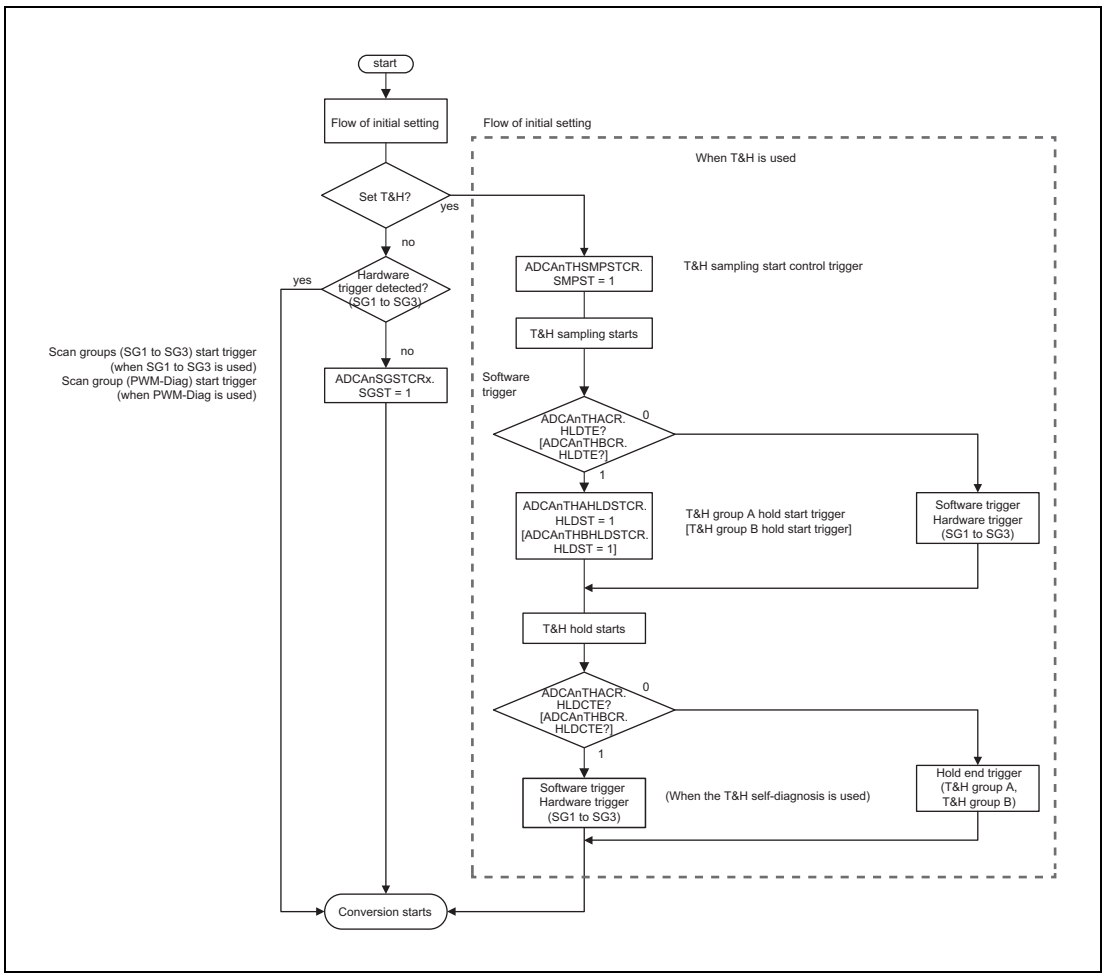


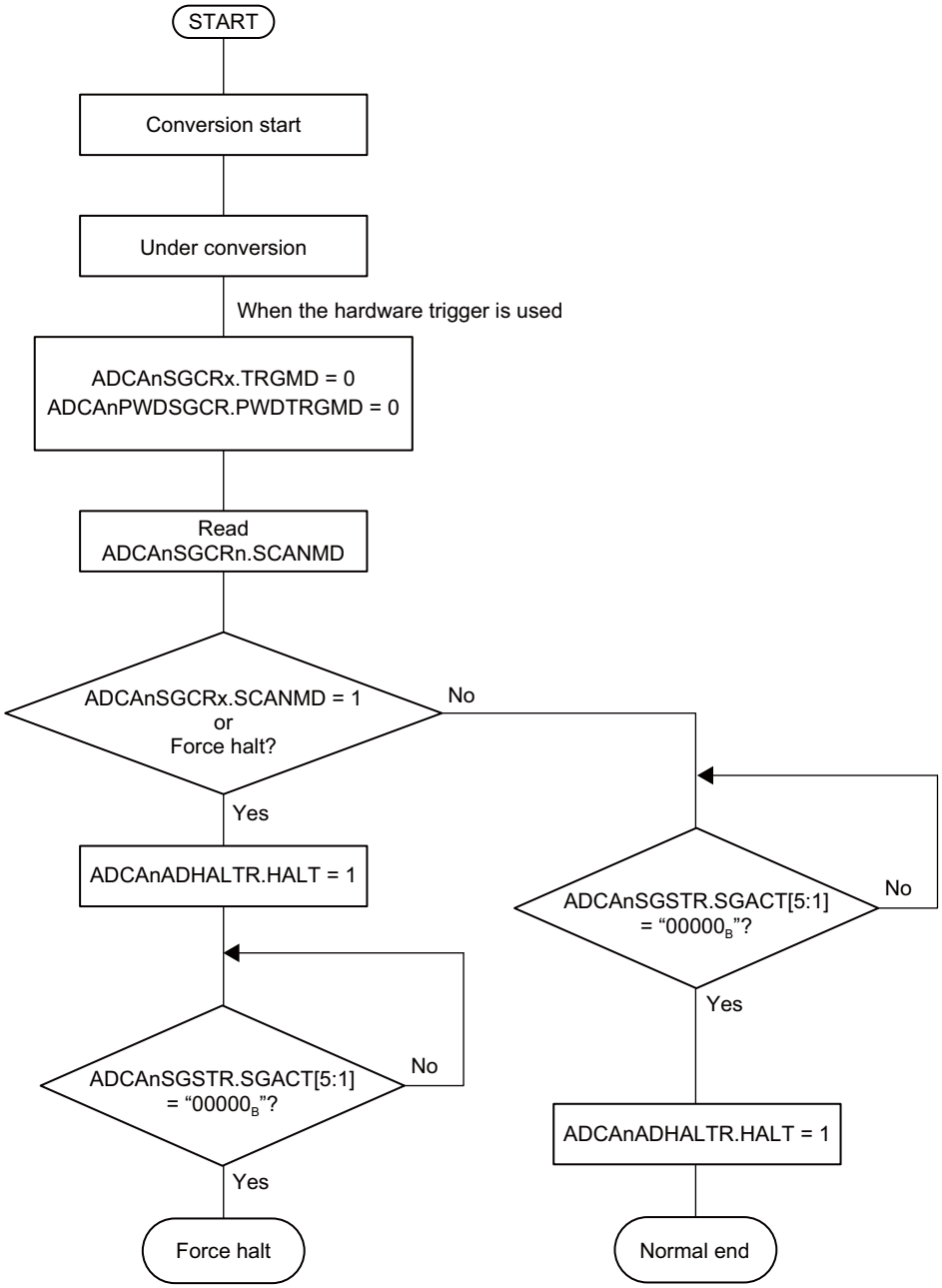

**Figure 31.7 Flowchart for Trigger Input**

**NOTE**

When an SG start trigger is generated during scanning, the SG start trigger is ignored.

### 31.4.3 Ending A/D Conversion

The flow for ending A/D conversion is shown below.



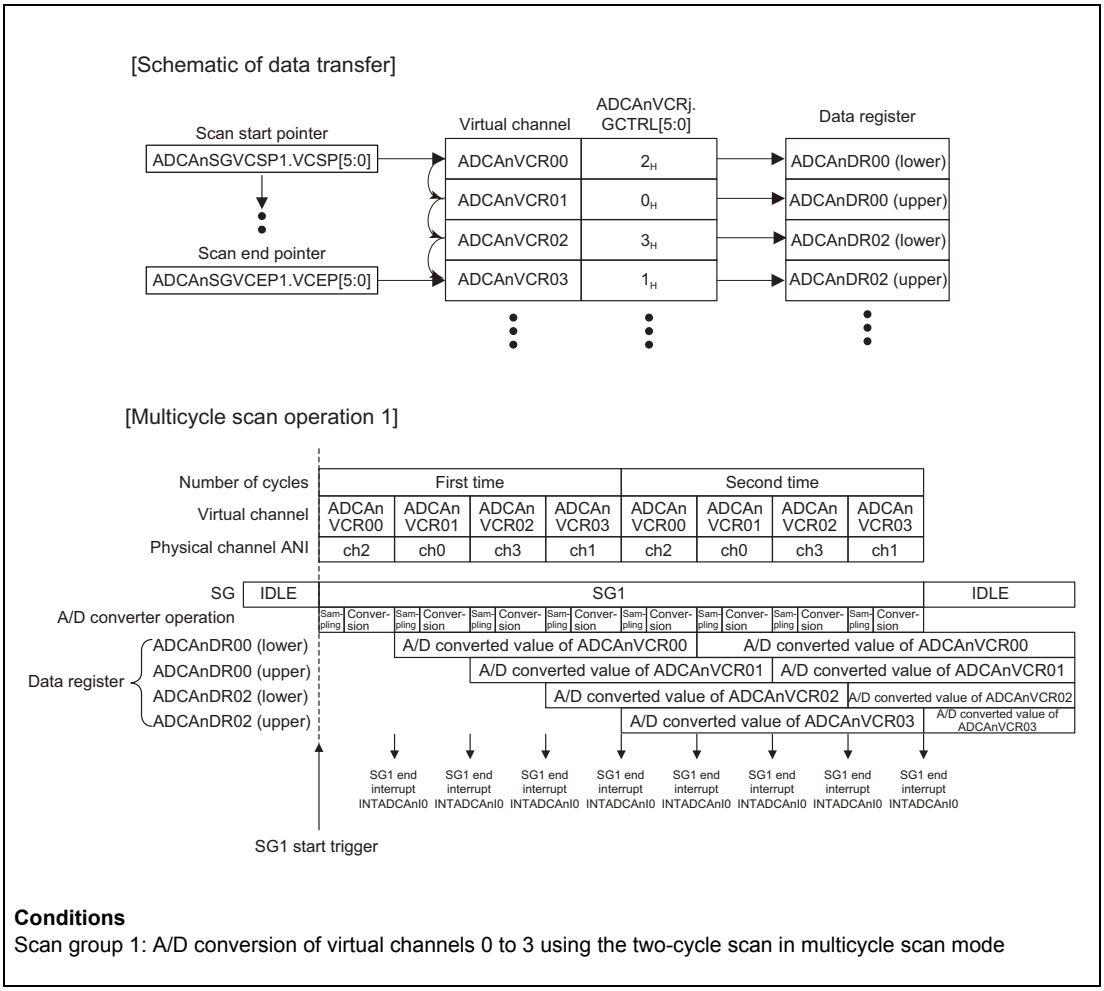

**Note:** The final setting of HALT when A/D conversion ends normally is done to clarify whether the status is "Conversion" or "IDLE". Whether HALT is set or not when A/D conversion ends normally does not effect the A/D conversion accuracy.

**Figure 31.8 Flowchart for Ending A/D Conversion**

## 31.4.4 Example of Scan Group Operation

### (1) Multicycle scan mode

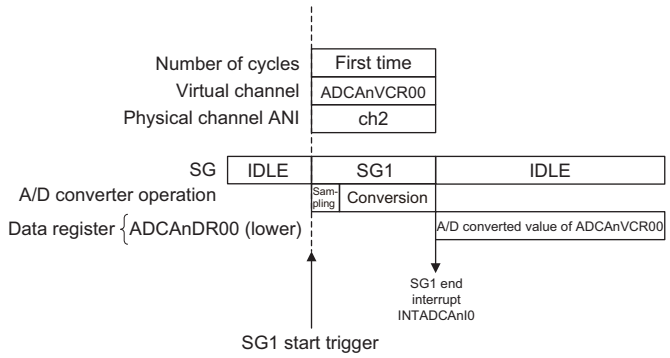
The following figure illustrates an operation example where four virtual channels of scan group 1 are converted using the two-cycle scan in multicycle scan mode.




**Conditions**
Scan group 1: A/D conversion of virtual channels 0 to 3 using the two-cycle scan in multicycle scan mode

**Figure 31.9 Example of Multicycle Scan Operation 1**

The following figure illustrates an operation example where a pin is scanned once in multicycle scan mode.



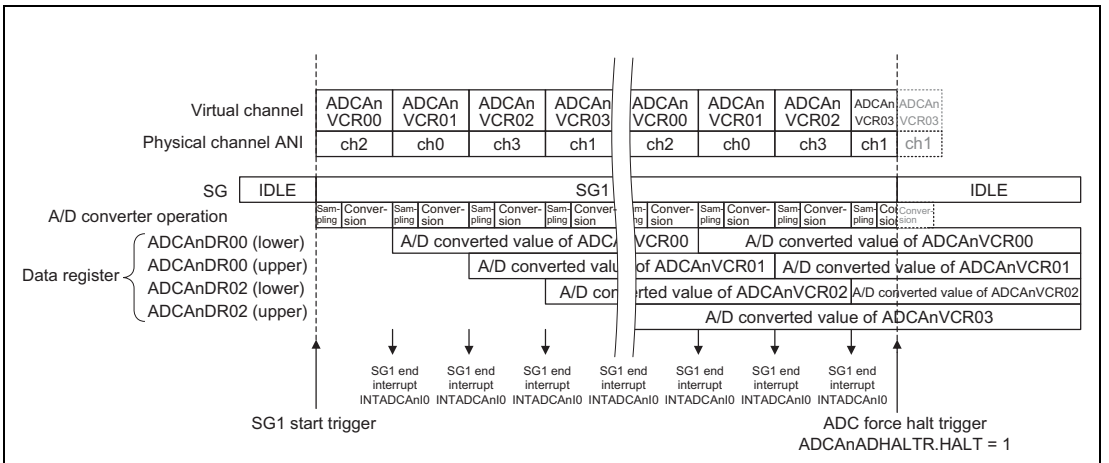

**Conditions**
Scan group 1: A/D conversion of virtual channel 0 in which a pin is scanned once in multicycle scan mode

**Figure 31.10 Example of Multicycle Scan Operation 2**

### (2) Continuous scan mode

Continuous scan mode allows A/D conversion of the SG channels indicated by the pointers specified by ADCAnSGVCSPx.VCSP[5:0] to ADCAnSGVCEPx.VCEP[5:0] to continue until ADCAnADHALTR.HALT is asserted. This mode can be used only with SG1.

The following figure shows an example of operation in continuous scan mode.




**Conditions**
Scan group 1: A/D conversion of virtual channels 0 to 3 in continuous scan mode

**Figure 31.11 Example of Continuous Scan Operation**

### 31.4.5 Channel Repeat Mode

Channel repeat mode allows A/D conversion of the SG channel indicated by the pointer specified by ADCAnSGVCSPx.VCSP[5:0] to ADCAnSGVCEPx.VCEP[5:0] to repeat number of channel repeat times specified by ADCAnSGCRx.SCT[1:0]. This mode operates exclusively in each SG. The number of channel repeat times is selectable from 1, 2, and 4.

The following figures show examples of operation under respective conditions.

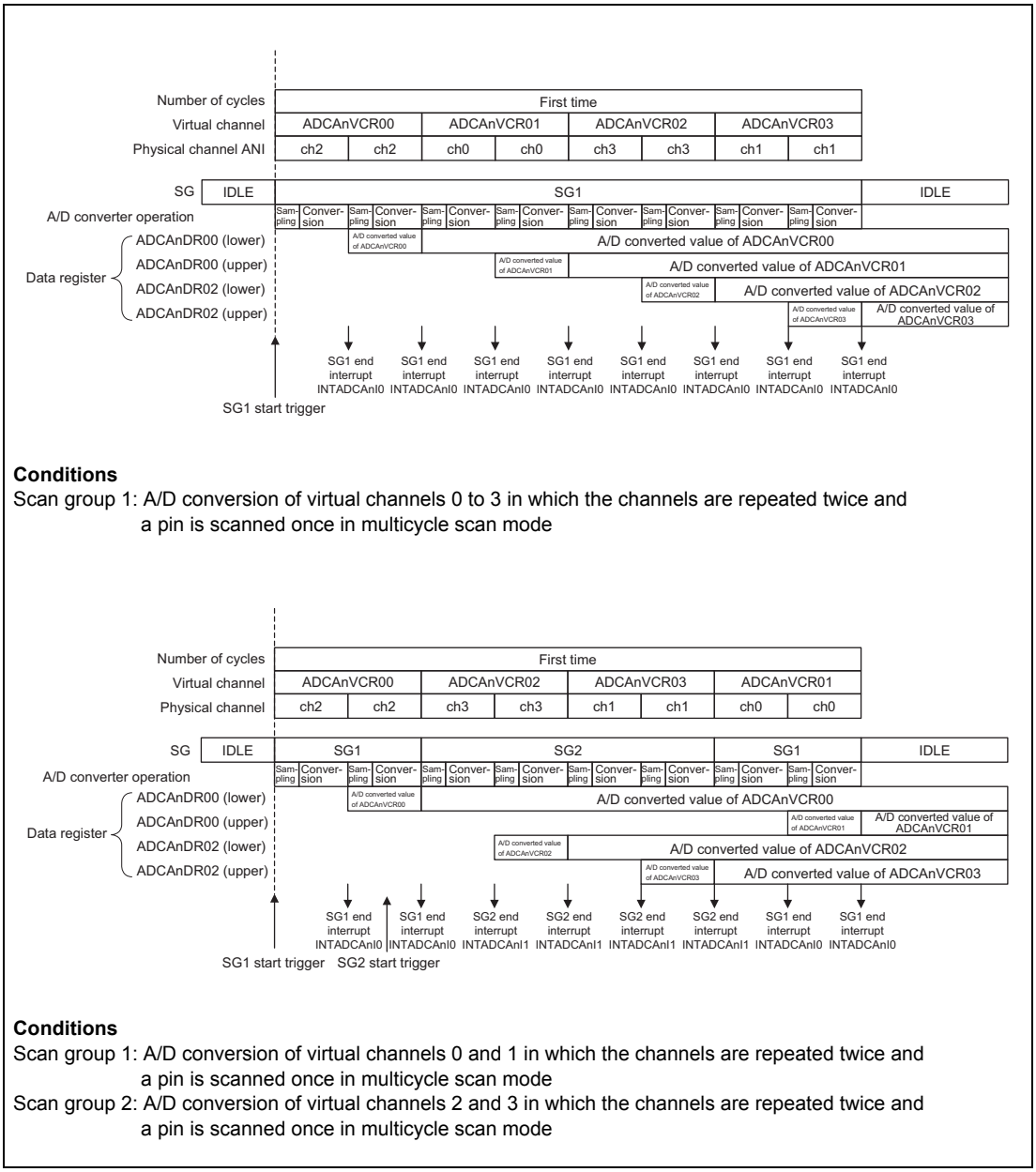
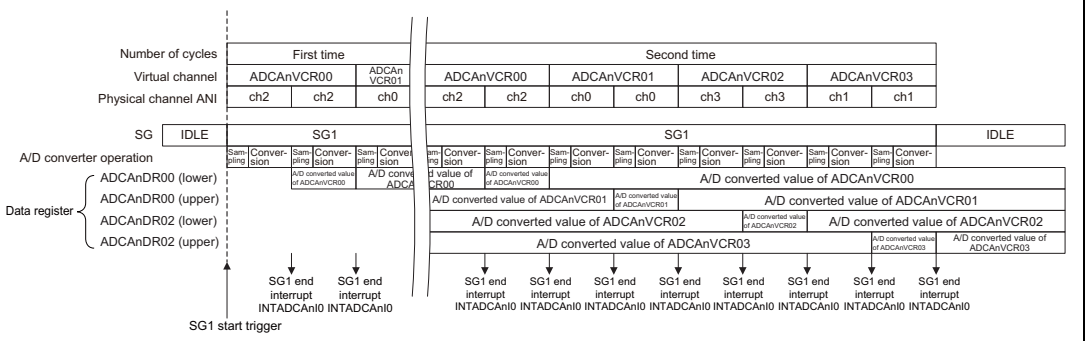


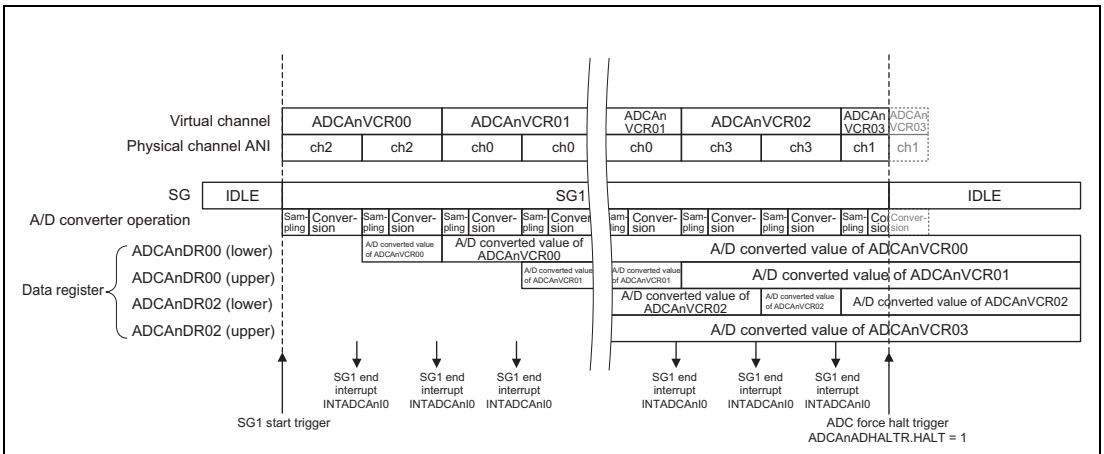

**Figure 31.12 Example of Channel Repeat Operation 1**




**Conditions**

Scan group 1: A/D conversion of virtual channels 0 to 3 in which the channels are repeated twice and the two-cycle scan is used in multicycle scan mode

**Figure 31.13 Example of Channel Repeat Operation 2**



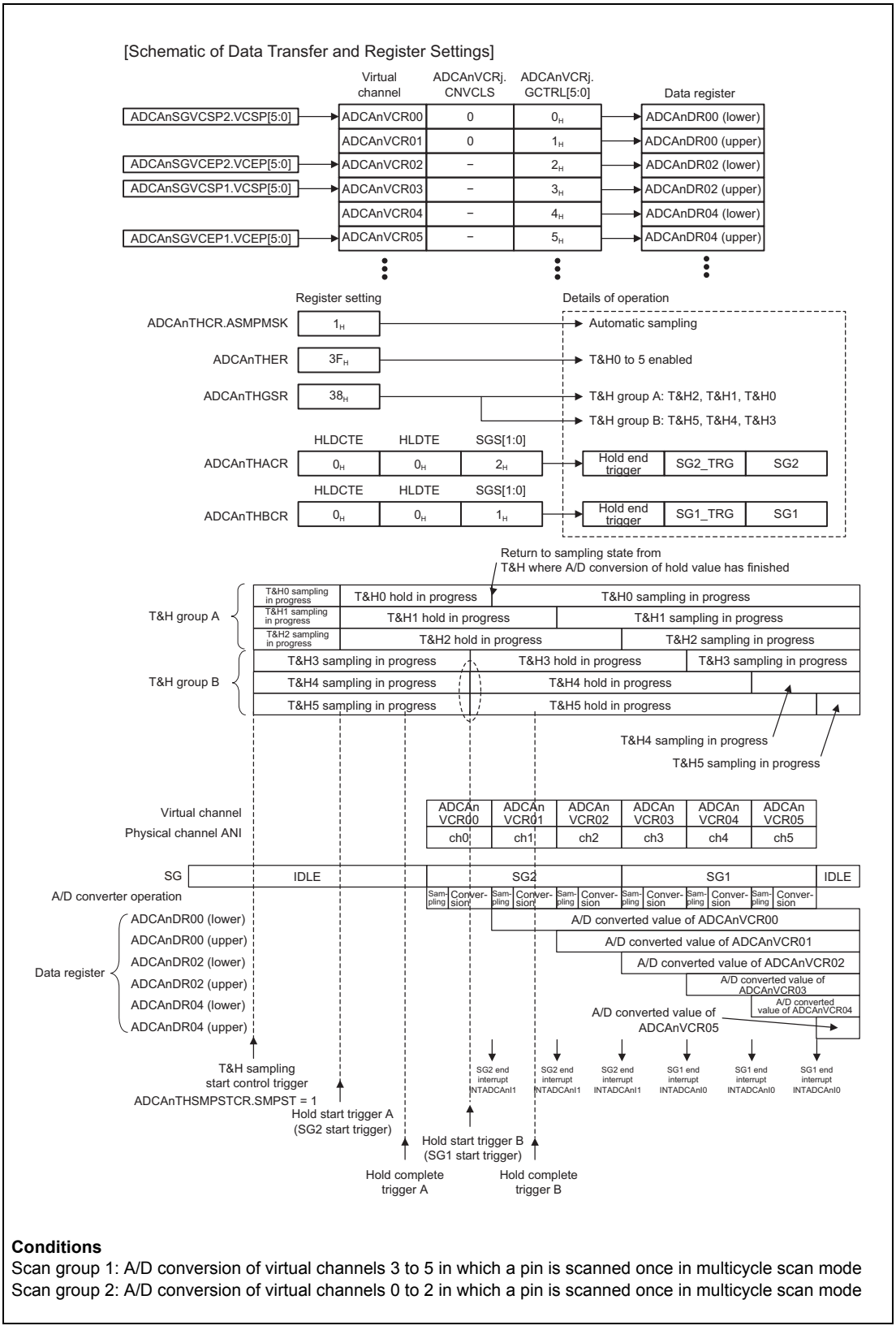

**Conditions**

Scan group 1: A/D conversion of virtual channels 0 to 3 in which the channels are repeated twice in continuous scan mode

**Figure 31.14 Example of Channel Repeat Operation 3**

### 31.4.6 Example of Simultaneous Track and Hold Operation

**Figure 31.15** shows an operation example of simultaneous track and hold.




**Conditions**
Scan group 1: A/D conversion of virtual channels 3 to 5 in which a pin is scanned once in multicycle scan mode
Scan group 2: A/D conversion of virtual channels 0 to 2 in which a pin is scanned once in multicycle scan mode

**Figure 31.15 Example of Simultaneous Track & Hold Operation 1**

**CAUTION**

- **Do not specify the same physical channel in different groups.**
- **Two-cycle (or more) scan in multicycle scan mode and track & hold operation using continuous scan mode are prohibited.**
- **Because ADCAnTHSMPSTCR.SMPST is common to group A and group B, set SMPST after T&H operation for both group A and group B has been completed.**
- **If the hardware trigger is asserted before HLDCTE = 1 and HLDTE = 1 of ADCAnTHACR register or ADCAnTHBCR register are set and HLDST of ADCAnTHAHLDSTCR register or ADCAnTHBHLDSTCR register is written to hold T&H, scan operation starts. In that case, the channel switch opens with T&H staying in the sampling state. Therefore, all scan results are undefined. Do not assert the hardware trigger by setting HLDCTE = 1 and HLDTE = 1 before writing HLDST.**
- **Setting any channel from among 0 to 2 and any channel from among 3 to 5 in the same scan group is prohibited.**
- **Set the interval between T&H sampling start control trigger and hold start trigger to be 450 ns or more.**
- **Set the interval between hold start trigger and completion of the group A/D conversion to be 10 µs or less.**
  **In suspend mode, do not use T&H for the channel of a scan group with low priority setting, if suspend time exceeds 10 µs.**

### 31.4.7 A/D Conversion with External Analog Multiplexer

The following figures show examples of A/D conversion in each case.

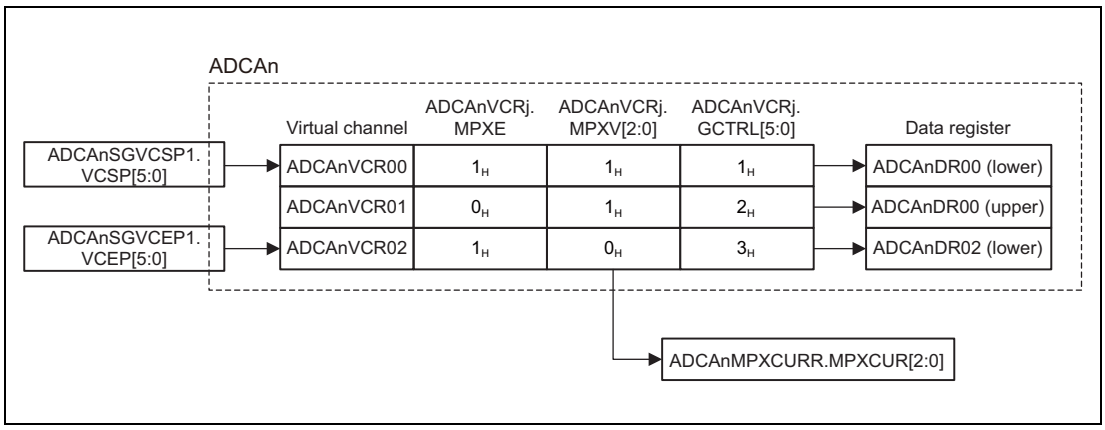
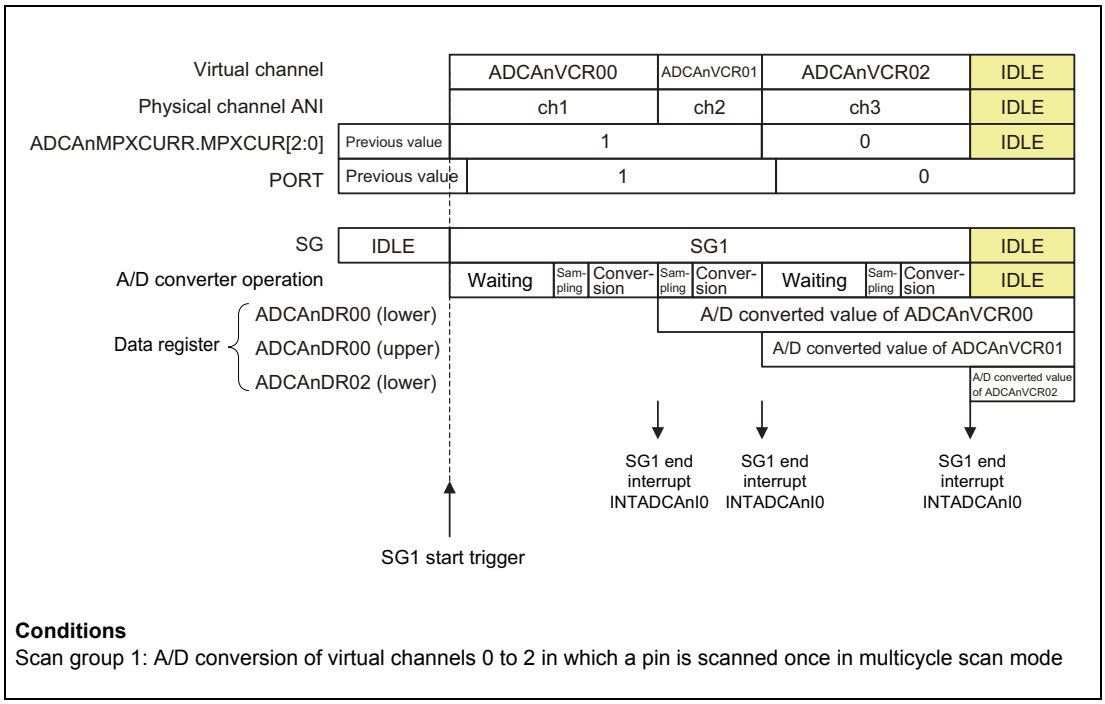
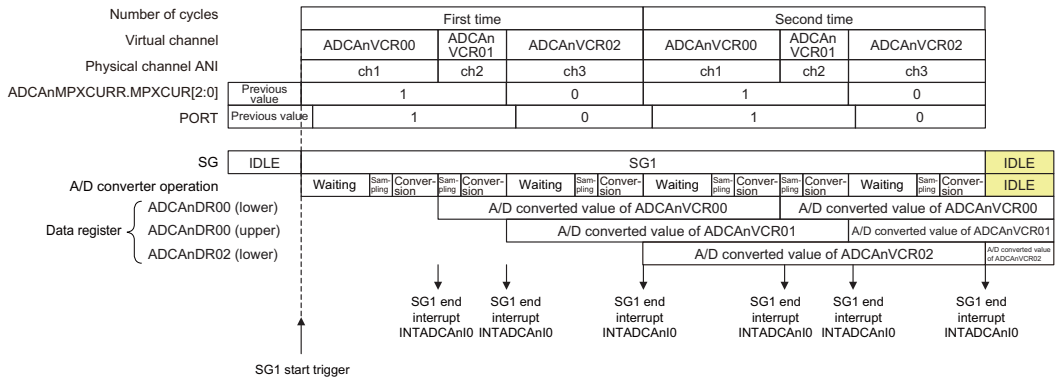



**Figure 31.16 Schematic of Data Transfer and Register Settings**




**Conditions**
Scan group 1: A/D conversion of virtual channels 0 to 2 in which a pin is scanned once in multicycle scan mode

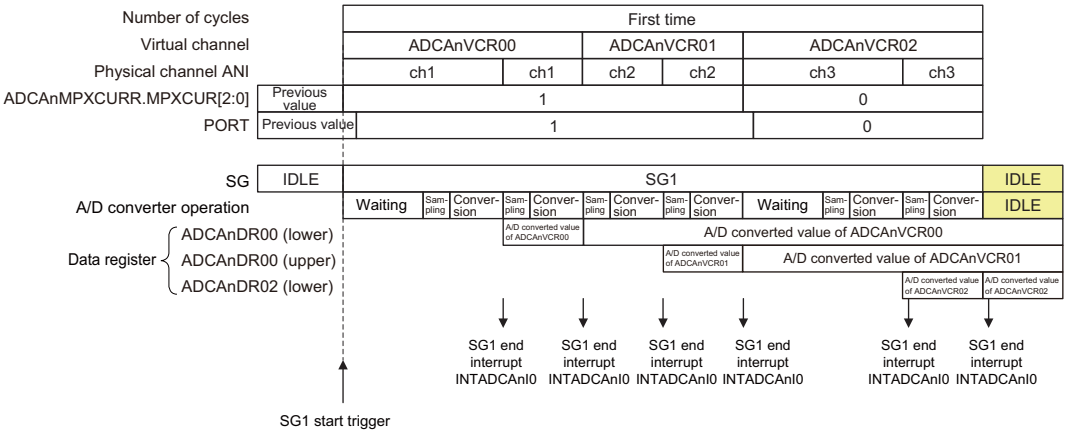
**Figure 31.17 A/D Conversion 1 at an External Analog Multiplexer**




**Conditions**
Scan group 1: A/D conversion of virtual channels 0 to 2 using the two-cycle scan in multicycle scan mode

**Note:** Waiting period is 1 A/D conversion time.

**Figure 31.18 A/D Conversion 2 at an External Analog Multiplexer**




**Conditions**
Scan group 1: A/D conversion of virtual channels 0 to 2 in which the channels are repeated twice and a pin is scanned once in multicycle scan mode

**Figure 31.19 A/D Conversion 3 at an External Analog Multiplexer**

#### 31.4.7.1 A/D Conversion with PWM-Diag Enabled

With the PWM-Diag function enabled, A/D conversion is performed by the signal from the PWM-Diag.

For details on the PWM-Diag function, see **Section 30, PWM Output/Diagnostic (PWM-Diag)**.

To control the A/D conversion, the A/D converter receives the setting information on the MPX by the A/D conversion trigger select (PWSA) signal. The flow of A/D conversion with PWM-Diag is as follows.

(1) Set the channel MPX value of the MPX to ADCAnPWDVCR.MPXV[2:0]. Up to 8 channels can be specified to the MPX.

(2) The A/D conversion is started by the trigger signal PWSA_ADTRG from the PWM-Diag. In addition, when the MPX enable bit (ADCAnPWDVCR.MPXE) is 1, a wait of one A/D-conversion time is inserted before A/D conversion is performed.

(3) At the end of A/D conversion, the scan end is notified to the PWM-Diag.

**CAUTION**

**As the trigger signal PWSA_ADTRG of PWM-Diag function has a higher-priority than SGx_TRG (x = 1 to 3), the operation of other scan groups may be kept waiting until the PWM-Diag function is ended.**

**Figure 31.20** shows an example of PWM-Diag operation using an MPX.

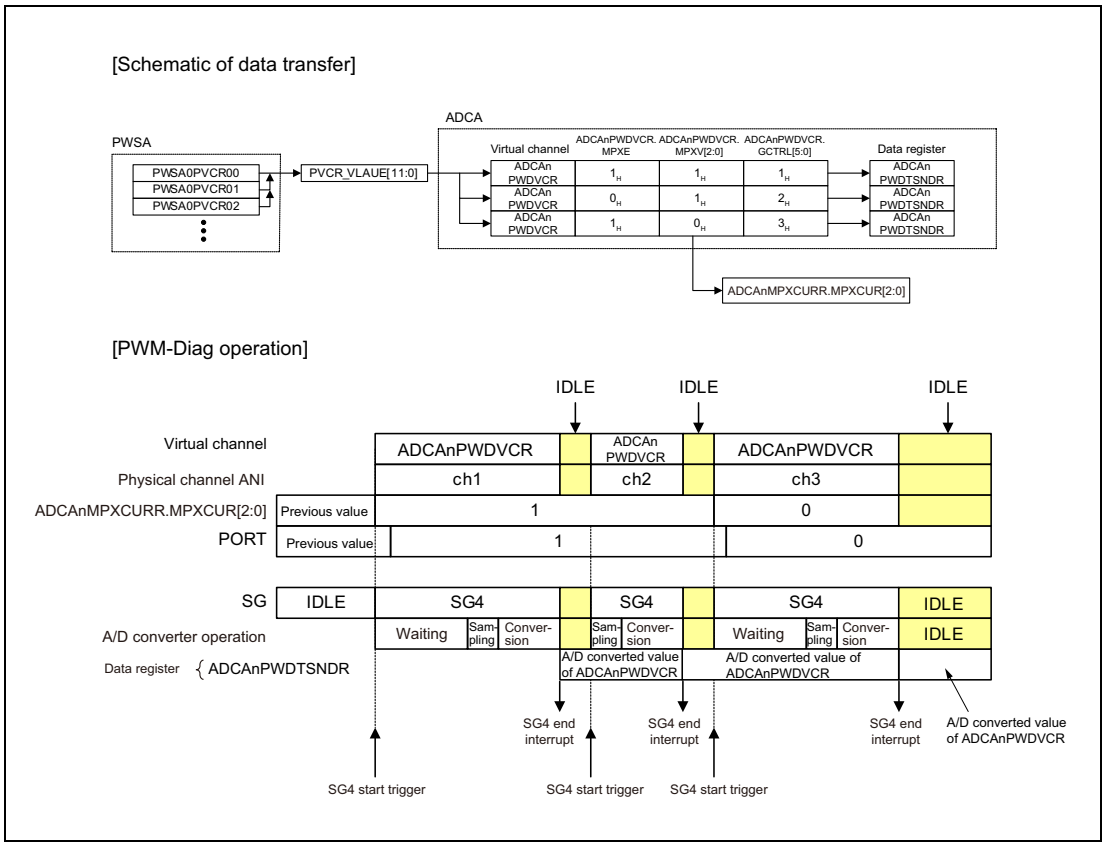



**Figure 31.20 PWM-Diag Operation**

### 31.4.8 Example of Synchronous Suspend and Resume Operation

**Figure 31.21** shows an example of synchronous suspend and resume operation. In this example, a higher-priority SG interrupts a lower-priority SG.

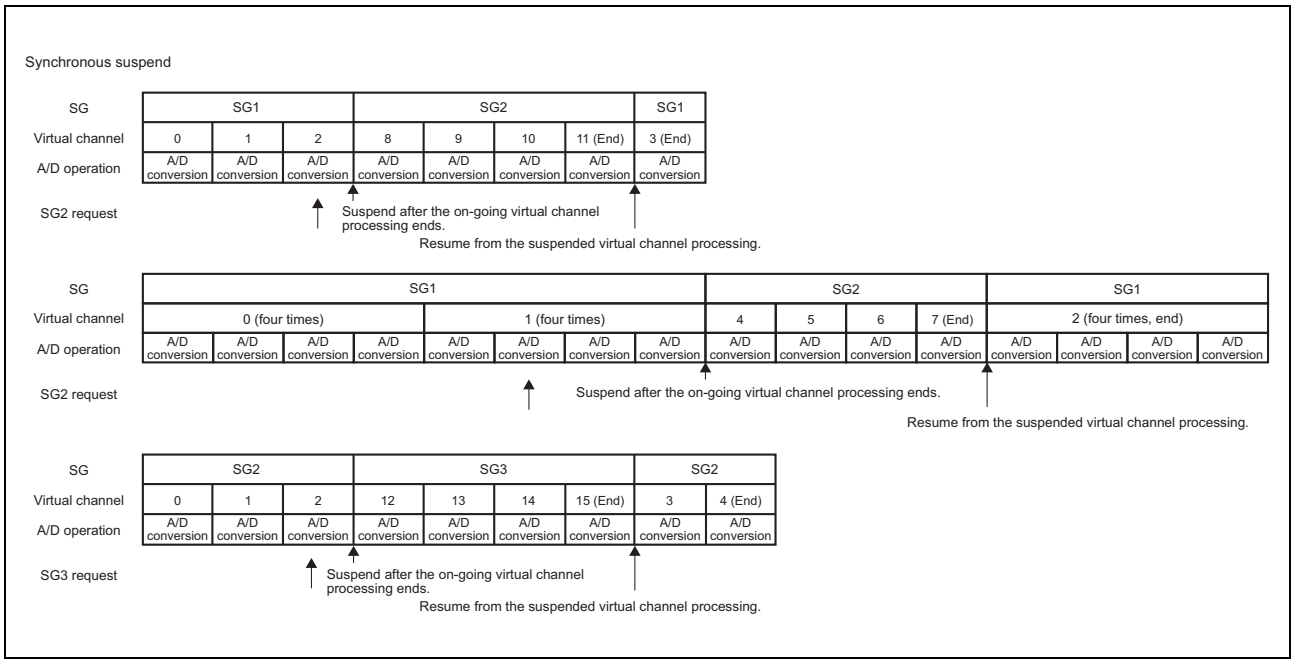



**Figure 31.21 Example of Synchronous Suspend and Resume Operation**

**NOTE**

Priority of scan groups is as follows.

Lower Higher

SG1 < SG2 < SG3 < PWM-Diag (SG4)

### 31.4.9 Example of Asynchronous Suspend and Resume Operation

**Figure 31.22** shows an example of asynchronous suspend and resume operation. In this example, a higher-priority SG interrupts a lower-priority SG.

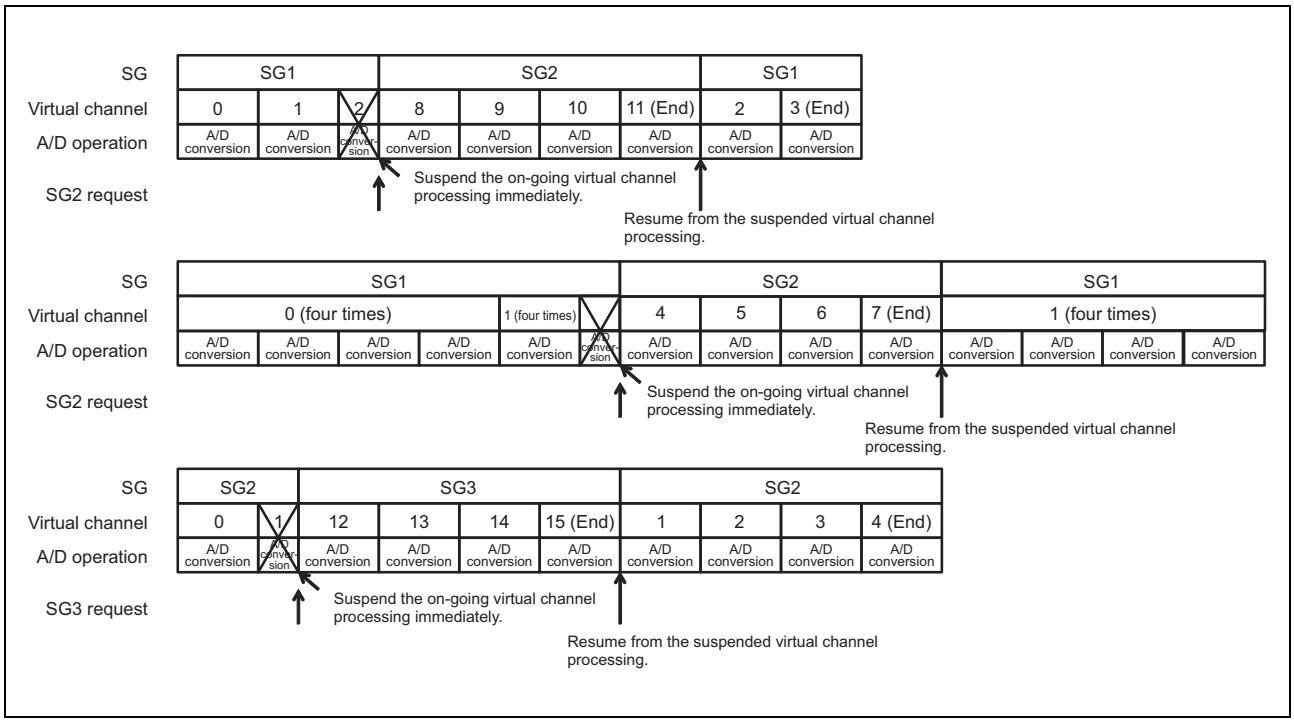



**Figure 31.22 Example of Asynchronous Suspend and Resume Operation**

**NOTE**

Priority of scan groups is as follows.

Lower Higher

SG1 < SG2 < SG3 < PWM-Diag (SG4)

## 31.4.10 Error Detecting Functions

ADCAn covers upper-limit error, lower-limit error, and overwrite error.

### 31.4.10.1 Upper-Limit/Lower-Limit Error Detecting Function

The upper-limit/lower-limit error detecting function determines whether the A/D converted data is larger than the upper-limit table ADCAnULLMTBR0.ULMTB[11:0] or smaller than the lower-limit table ADCAnULLMTBR0.LLMTB[11:0] at the end of A/D conversion.

### 31.4.10.2 Overwrite Error Detecting Function

If the ADCAnDIRj register or ADCAnDRj register of a virtual channel is not read while ADCAnDIRj.WFLG = 1 (A/D converted value is stored) and the next A/D converted value is written in the ADCAnDRj register, an overwrite error is detected.

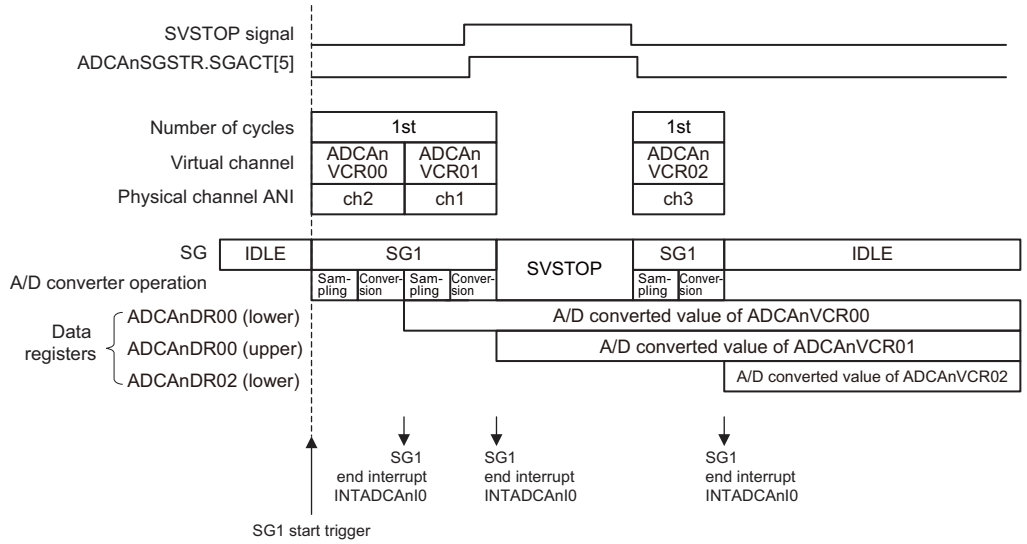
### 31.4.10.3 SVSTOP Operation

The SVSTOP function is supported by the SVSTOP signal sent from the on-chip debugger control unit. The SVSTOP function stops conversion of the A/D converter when the SVSTOP signal is input during an emulation break. While the SVSTOP signal is high, reading registers ADCAnDRj, ADCAnDIRj, ADCAnSGSTR, ADCAnULER, ADCAnOWER, ADCAnPWDTSNDR, and ADCAnPWDDIR by external access does not affect these registers.

When the high level is input to SVSTOP while ADCAnEMU.SVSDIS = 0, ADCAnSGSTR.SGACT[5] is set to 1 to make a transition to the SVSTOP state. Hardware triggers and software triggers are valid in the SVSTOP state. When the high level is input to SVSTOP while ADCAnEMU.SVSDIS = 1, the ADCA does not make a transition to the SVSTOP state. ADHALT (forced termination of A/D conversion) should not be performed in the SVSTOP state.

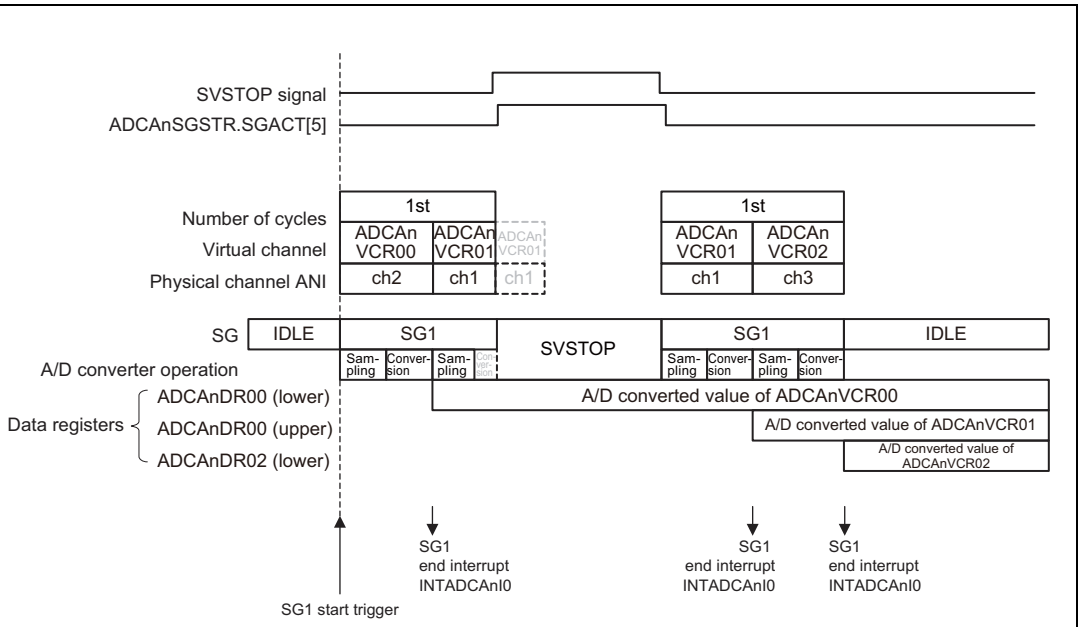
In operations for synchronous suspension, a new start trigger cannot be accepted over the time from when the high level is input to SVSTOP to the completion of conversion on the channel where conversion is currently proceeding. This time can be up to the time taken for one A/D conversion.

The following example illustrates a SVSTOP operation example.



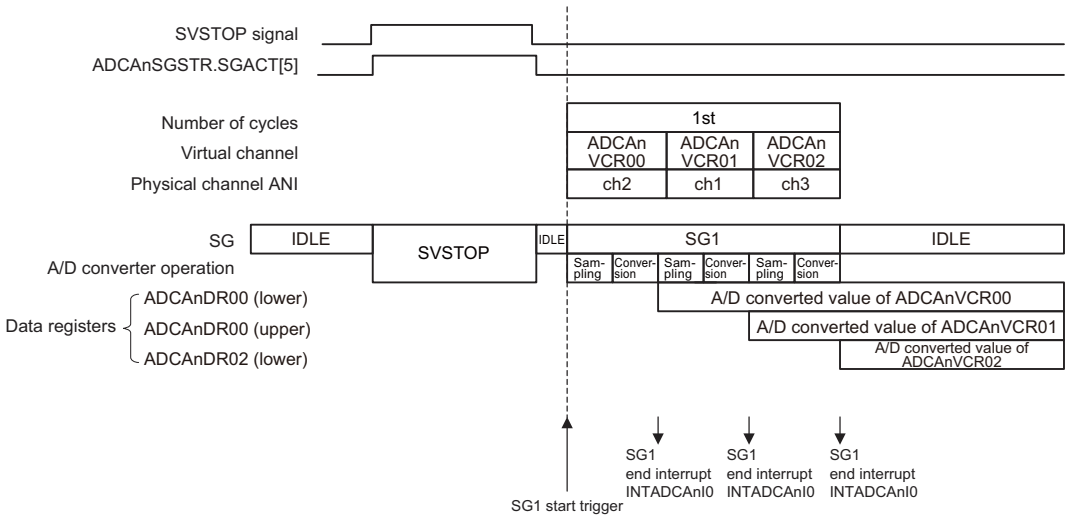

**Note:** An SG start trigger is accepted even during SVSTOP after the completion of conversion.

**Figure 31.23 Example of SVSTOP Operation (ADCAnADCR.SUSMTD = 00 and ADCAnEMU.SVSDIS = 0)**



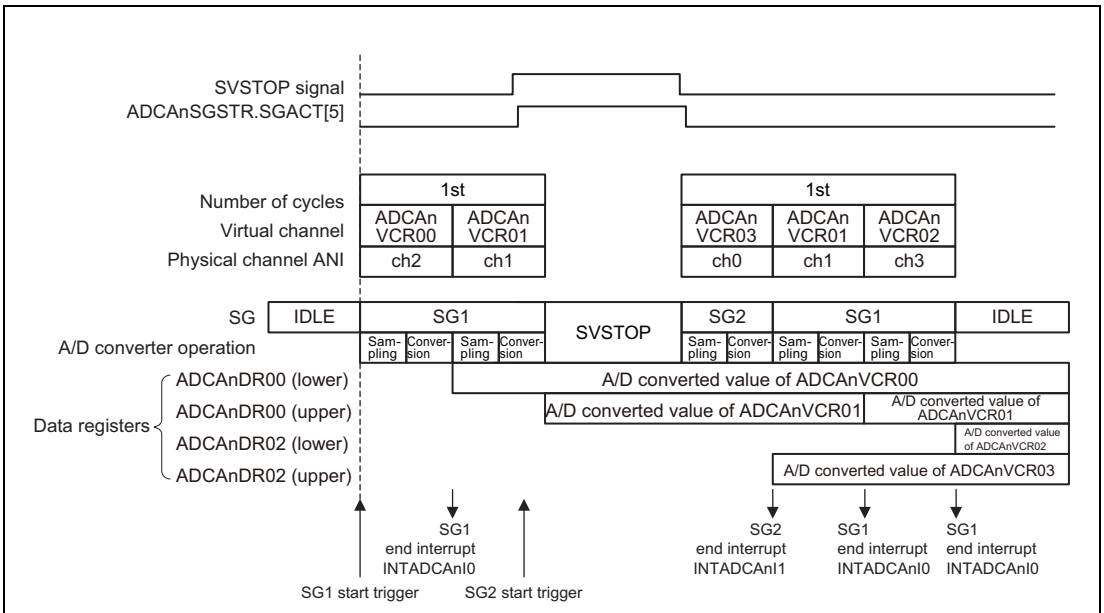

**Note:** An SG start trigger is accepted even during SVSTOP.

**Figure 31.24 Example of SVSTOP Operation (ADCAnADCR.SUSMTD = 10 and ADCAnEMU.SVSDIS = 0)**



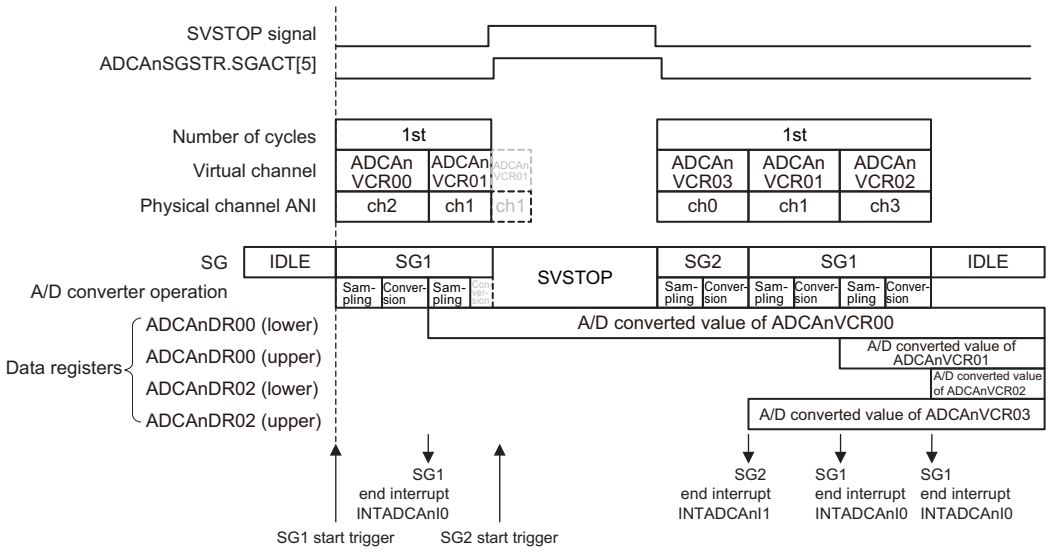

**Note:** An SG start trigger is accepted even during SVSTOP.

**Figure 31.25 Example of SVSTOP Operation in the IDLE State (ADCAnADCR.SUSMTD = 00 and ADCAnEMU.SVSDIS = 0)**



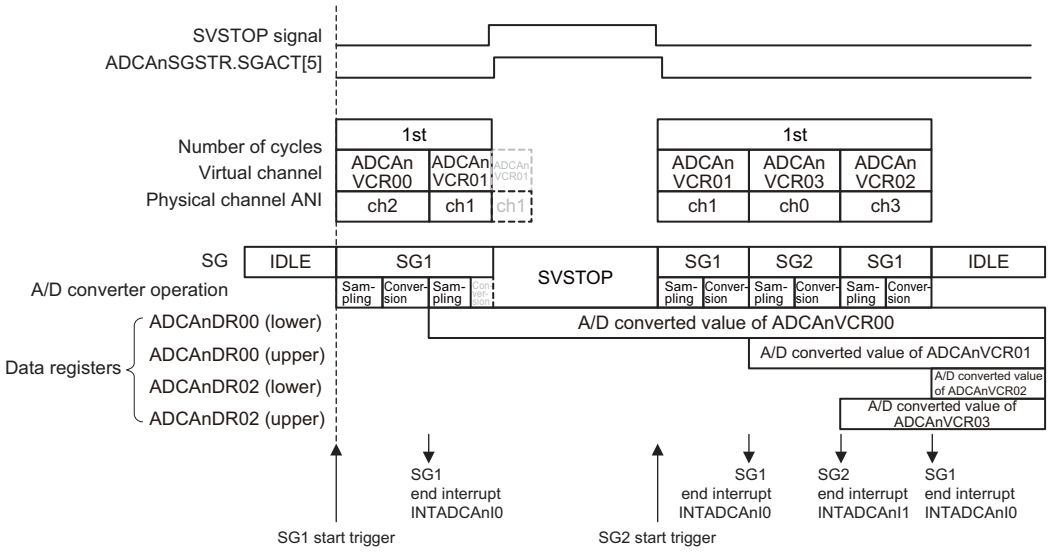

**Note:** Though the channel where conversion is currently proceeding is not stopped even after the acceptance of SVSTOP, an SG end interrupt is not generated.
This channel is converted again after SVSTOP has been cleared.
After the completion of conversion, the SG start trigger is accepted even during SVSTOP.

**Figure 31.26 Conflict of SVSTOP Start and High-Priority SG Start Trigger (ADCAnADCR.SUSMTD = 00, ADCAnEMU.SVSDIS = 0)**




**Note:** An SG start trigger is accepted even during SVSTOP.

**Figure 31.27 Conflict of SVSTOP Start and High-Priority SG Start Trigger (ADCAnADCR.SUSMTD = 10, ADCAnEMU.SVSDIS = 0)**




**Note:** An SG start trigger is accepted even during SVSTOP.
In cases of conflict between clearing of SVSTOP and setting of the scan group start trigger bit, conversion of the suspended scan group is resumed regardless of its priority.

**Figure 31.28 Conflict of SVSTOP Clear and High-Priority SG Start Trigger (ADCAnADCR.SUSMTD = 10, ADCAnEMU.SVSDIS = 0)**

### 31.4.11 Activating Scan Group by a Hardware Trigger

Scan group x can be activated by the hardware trigger input to SGx_TRG. As for the hardware trigger sources to be used, see **Table 31.49, List of A/D Conversion Hardware Triggers**. When activating SGx_TRG by the hardware trigger, set the peripheral function to be used by the trigger and set the start trigger in the A/D conversion trigger select control register (ADCAnSGTSELx).
A hardware trigger from external trigger input pin requires digital filter setting. For details, see **Section 2.12, Noise Filter & Edge/Level Detector**. More than one start trigger can be specified.

#### 31.4.11.1 Stopping Scan Group by ADHALT

Setting ADCAnADHALTR.HALT (A/D force halt trigger) to 1 forcibly halts the A/D conversion and clears the scan group status register (ADCAnSGSTR). The error flag of ADCAnULER (upper limit/ lower limit error register) is not cleared. When ADCAnADHALTR.HALT is set, make sure that ADCAnSGSTR.SGACT has been cleared.

### 31.4.12 Scan End Interrupt Request

Scan group x can issue a scan end interrupt request (INT_SGx) to INTC. If ADIE of ADCAnSGCRx is set to 1, INT_SGx can be output after the SGx scan ends. If ADIE of ADCAnSGCRx is set to 0, the INT_SGx output when the SGx scan ends can be disabled. If ADIE of ADCAnVCRj is set to 1, INT_SGx can be output when A/D conversion for virtual channel j in SGx ends. If ADIE of ADCAnVCRj is set to 0, the INT_SGx output when A/D conversion for virtual channel j in SGx ends can be disabled. Since SGx scan ending is simultaneous with A/D conversion ending for virtual channel j in SGx when ADIEs of both ADCAnSGCRx and ADCAnVCRj are set to 1, INT_SGx occurs only once.

Example 1: A scan is executed for virtual channel 0 or 1 in SG1 when ADIE of ADCAnSGCR1 is 0, ADIE of VCR0 is 1, and ADIE of VCR1 is 0.
INT_SG1 is output when A/D conversion ends for virtual channel 0.

Example 2: A scan is executed for virtual channel 0 or 1 in SG2 when ADIE of ADCAnSGCR2 is 0, ADIE of VCR0 is 1, and ADIE of VCR1 is 1.
INT_SG2 is output when A/D conversion ends for virtual channel 0 and virtual channel 1.

Example 3: A scan is executed for virtual channel 0 or 1 in SG3 when ADIE of ADCAnSGCR3 is 1, ADIE of VCR0 is 0, and ADIE of VCR1 is 0.
INT_SG3 is output when a scan ends (when A/D conversion for virtual channel 1 ends).

Furthermore, the DMAC can be started when scan ends.

For the setting of DMAC, see **Section 8, DMA Controller**.

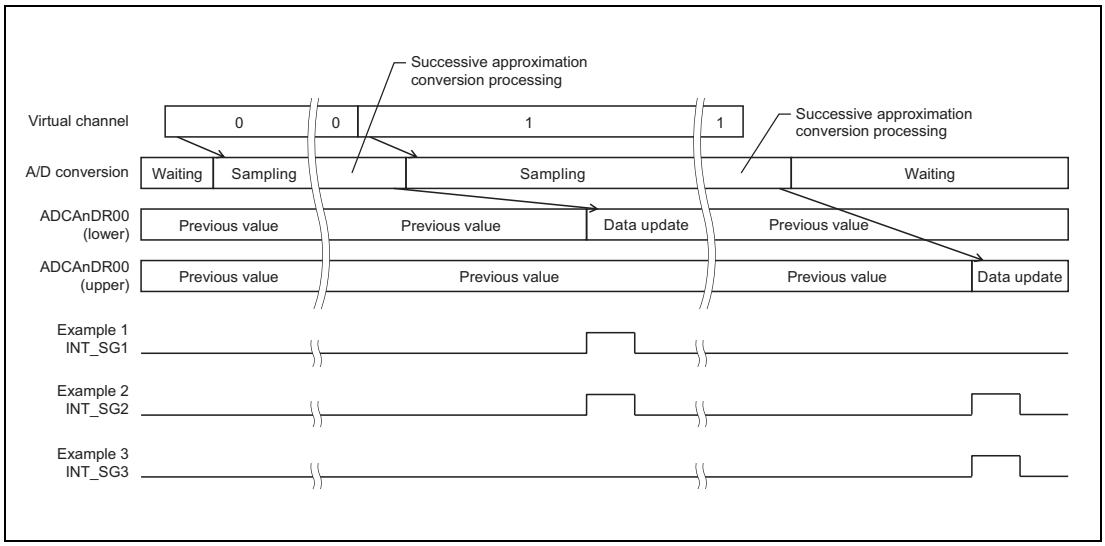



**Figure 31.29 Example of a Scan Conversion End Interrupt Occurrence**

### 31.4.13 A/D Error Interrupt Request

ADCA can issue an A/D error interrupt request (INT_ADE) to INTC. For an error source for which ULEIE and OWEIE of ADCAnSFTCR are set to 1, the OR condition of the error source is issued as INT_ADE. For an error source for which ULEIE and OWEIE of ADCAnSFTCR are set to 0, INT_ADE does not output an interrupt.

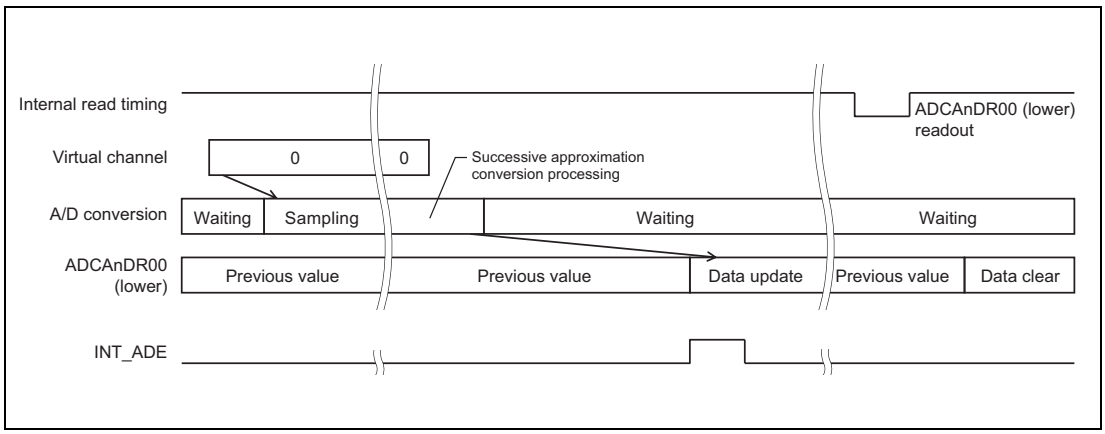



**Figure 31.30 A/D Error Interrupt (Example: Overwrite Error)**

## 31.5 Self-Diagnostic Function

To check the ADCAn function, the following self-diagnostic functions are available.

**Section 31.5.1, Diagnosis of A/D Conversion Circuit**

**Section 31.5.2, Diagnosis of Channel Multiplexer**

**Section 31.5.3, Diagnosis of Open Pins**

**Section 31.5.4, Diagnosis of T&H Circuit**

The overview of the self-diagnostic functions is shown in the figure below. A detailed description is given in the following sections.

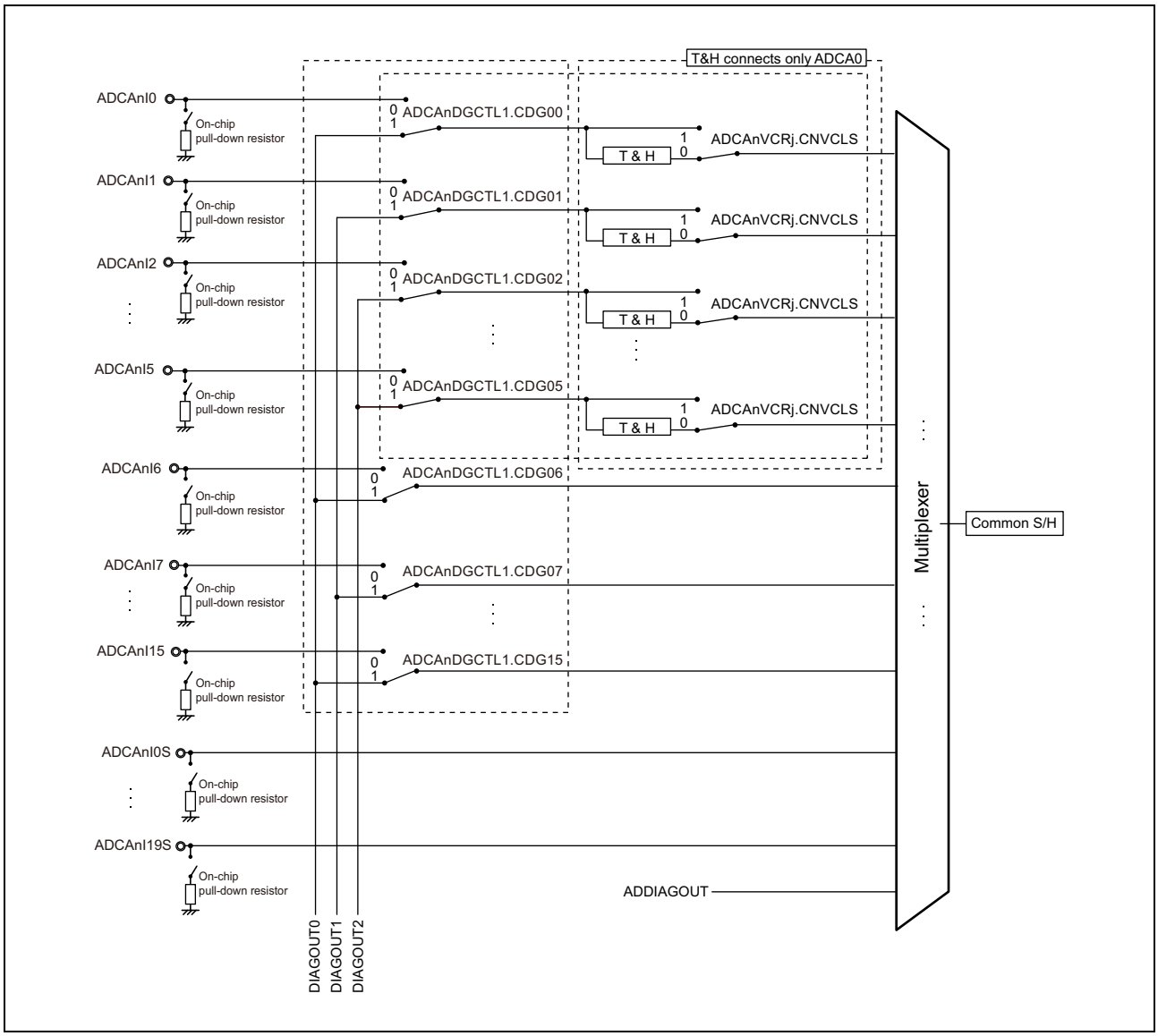



**Figure 31.31 Overview of Self-Diagnostic Functions**

**NOTE**

The functions in the dashed-line frames depend on the product.

### 31.5.1 Diagnosis of A/D Conversion Circuit

This function checks whether the A/D converter is operating normally by verifying the A/D conversion for self-diagnostic voltage (ADDIAGOUT) and the result of conversion. If the result of A/D conversion differs from the expected value, an internal circuit may be broken. The features of self-diagnosis of the A/D converter are as follows:

- As the self-diagnostic voltage (ADDIAGOUT) level, $AnV_{REF}$, $2/3AnV_{REF}$, $1/3AnV_{REF}$, $1/2AnV_{REF}$, and AnVSS are selectable by the PSEL[2:0] bits in the ADCAnDGCTL0 register.
- Self-diagnosis of the A/D converter is enabled by performing A/D conversion on one of SG1 to SG3.

#### 31.5.1.1 Diagnostic procedure

The diagnostic procedures are shown below.

Common settings for ADC should be made before self-diagnosis is to proceed.

1. Set ADCAnADCR.DGON = 1 to enable the self-diagnostic voltage circuit.
2. Wait for 500 ns.
3. Set ADCAnDGCTL0.PSEL[2:0] to select a self-diagnostic voltage level.
4. Set ADCAnADCR.DGON = 1 to update the voltage level.
5. Wait for 500 ns.
6. Set an arbitrary bit of ADCAnVCRj.GCTRL[5:0] to $100100_B$ to select the diagnosis channel.
7. Set ADCAnVCRj.ADIE = 1 to enable the A/D conversion end interrupt.
8. Set ADCAnSGVCSPx to specify the start pointer of virtual channel.
9. Set ADCAnSGVCEPx to specify the end pointer of virtual channel.
10. Generate the start trigger of scan group to perform the A/D conversion.
11. When the conversion interrupt occurs, read the result and compare it with the expected one.
12. If the result is the expected one, the A/D conversion was performed successfully.

**NOTE**

- During A/D conversion, the self-diagnostic voltage level can be changed by writing to ADCAnDGCTL0.PSEL[2:0]. However, the value of ADCAnDGCTL0.PSEL[2:0] becomes effective from the next A/D conversion.
- To clear ADCAnADCR.DGON, follow the procedure below:
  1. Confirm that SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.
  2. Clear ADCAnDGCTL0.PSEL[2:0].
  3. Clear ADCAnADCR.DGON.

### 31.5.2 Diagnosis of Channel Multiplexer

This function checks whether the path from the analog input to the A/D converter is normal.
Set the A/D conversion reference voltage (DIAGOUT0, DIAGOUT1, DIAGOUT2) by ADCAnDGCTL0.PSEL[2:0] and the channels to be connected by the ADCAnDGCTL1 register to perform A/D conversion using multiple analog channels.
If the result of A/D conversion differs from the expected value, an internal circuit may be broken. The features of self-diagnosis of the channel multiplexer are as follows:

- Channels for diagnosis can be arbitrarily selected from among ADCA0I0 to ADCA0I15 and ADCA1I0 to ADCA1I15.
- As the self-diagnostic voltage level, $2/3AnV_{REF}$, $1/3AnV_{REF}$, and $1/2AnV_{REF}$ are selectable and one of the three reference voltage levels can be allocated to each channel.

**Table 31.55 Selection of Channel to be Diagnosed**

| Connection | Select Channel |
|---|---|
| DIAGOUT0 | Channels 0, 3, 6, 9, 12, and 15 |
| DIAGOUT1 | Channels 1, 4, 7, 10, and 13 |
| DIAGOUT2 | Channels 2, 5, 8, 11, and 14 |

- Self-diagnosis of the channel multiplexer is enabled by performing A/D conversion on one of SG1 to SG3 by using multiple channels.

#### 31.5.2.1 Diagnostic procedure

The diagnostic procedure is shown below.

Common settings for ADC should be made before self-diagnosis is to proceed.

1. Set ADCAnADCR.DGON = 1 to enable the self-diagnostic voltage circuit.
2. Wait for 500 ns.
3. Set ADCAnDGCTL0.PSEL[2:0] to select a self-diagnostic voltage level.
4. Set ADCAnADCR.DGON = 1 to update the voltage level.
5. Wait for 500 ns.
6. Use two or more ADCAnVCRj registers.
   Set ADCAnVCRj.GCTRL[5:0] bits to select physical channels.
   Set ADCAnVCRj.ADIE bit to enable the A/D conversion end interrupt.
7. Set ADCAnSGVCSPx register to specify the start pointer of virtual channel.
8. Set ADCAnSGVCEPx register to specify the end pointer of virtual channel.
9. Set ADCAnDGCTL1 register to specify the physical channel to the self-diagnostic channel.
10. Generate the start trigger of scan group to perform the A/D conversion.
11. When the conversion interrupt occurs, read the result and compare it with the expected one.
12. If the result is the expected one, the A/D conversion was performed successfully.

**NOTE**

- During A/D conversion, the self-diagnostic voltage level can be changed by writing to ADCAnDGCTL0.PSEL[2:0]. However, the value of ADCAnDGCTL0.PSEL[2:0] becomes effective from the next A/D conversion.
- To clear ADCAnADCR.DGON, follow the procedure below:
  1. Confirm that SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.
  2. Clear ADCAnDGCTL0.PSEL[2:0].
  3. Clear ADCAnADCR.DGON.

### 31.5.3 Diagnosis of Open Pins

This function detects whether the analog input pin (ADCAnIm, ADCAnImS) is open due to disconnection, etc.
An internal pull-down resistor can be connected to diagnose the analog input pin.
Connect the analog input pin (ADCAnIm, ADCAnImS) with the pull-down resistor for self-diagnosis for A/D conversion of the target channels.

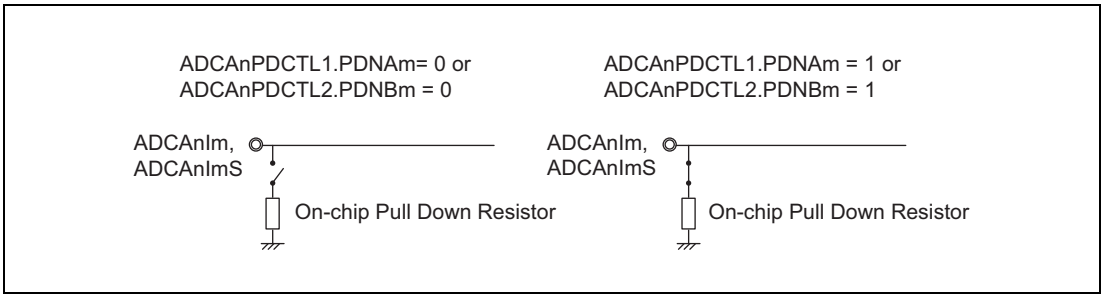



**Figure 31.32 Setting of On-chip Pull Down Resistor**

When there is a disconnection, the conversion result is almost 0 V and it indicates an open detection.

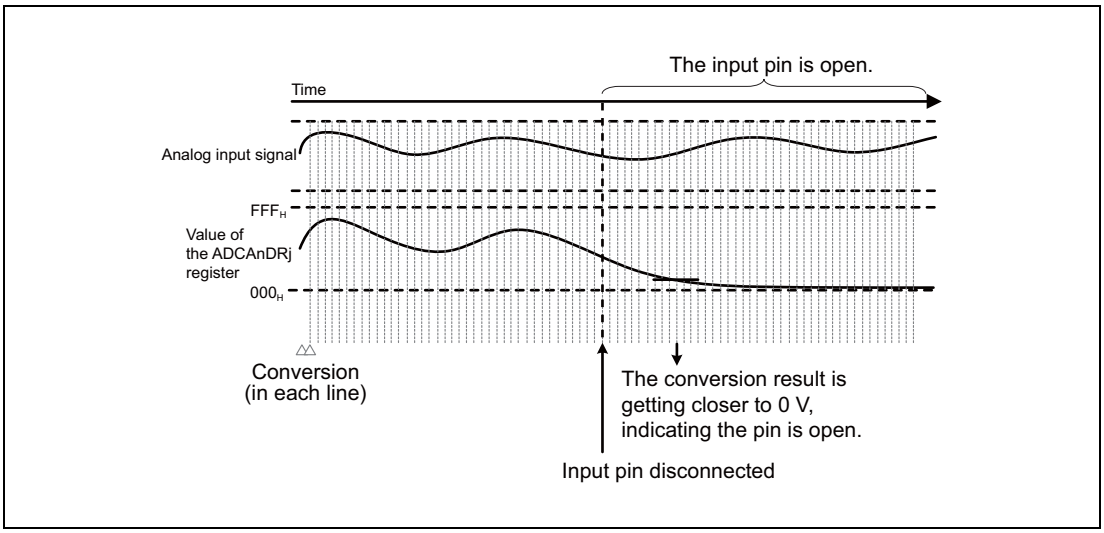



**Figure 31.33 Analog Input Signal Disconnection Detection**

**CAUTIONS**

1. **The pull-down resistors must not be connected during normal A/D conversion operation. Connected pull-down resistors may lead to a drop in the input voltage and result in erroneous A/D conversion results.**
2. **When the analog input voltage is nearly equal to the voltage level which is pulled down, a disconnection cannot be detected by this function.**

#### 31.5.3.1 Diagnostic procedure

1. Set the ADCAnPDCTL1.PDNA or ADCAnPDCTL2.PDNB bit which correspond to analog input pins (ADCAnIm, ADCAnImS) to be diagnosed to enable the pull down resistor.
2. Generate the start trigger of scan group to perform the A/D conversion.
3. Perform the A/D conversion multiple times on the same analog input.
4. Monitor the channel's A/D conversion results and check if any result declines to 0 V.

### 31.5.4 Diagnosis of T&H Circuit

This function is used to diagnose proper operation of the T&H0 to T&H5 circuits for ADCA0I0 to ADCA0I5.

Virtual channel registers 33 to 35 (ADCA0VCR33 to 35) are used exclusively for comparison of the potential conversion result using the T&H circuit and that obtained without using the T&H circuit to detect a failure of the T&H circuit.

For this diagnosis, the ADCA0THACR.HLDCTE is set to 1 (ADCA0THBCR.HLDCTE = 1) and ADCA0THACR.HLDTE to 0 (ADCA0THBCR.HLDTE = 0) and the A/D conversion trigger is used as the hold start/end trigger. Connect the reference voltage signal (DIAGOUT0, DIAGOUT1, or DIAGOUT2) selected by ADCAnDGCTL0.PSEL[2:0] to the target channels for diagnosis by using the ADCAnDGCTL1 register.

#### 31.5.4.1 Diagnostic Procedure (in case of T&H circuit ch0 diagnosis)

1. Set ADCA0ADCR.DGON = 1 to enable the self-diagnostic voltage circuit.
2. Wait for 500 ns.
3. Set ADCA0DGCTL0.PSEL[2:0] = $001_B$ to select $1/3AnV_{REF}$ voltage level.
4. Set ADCA0ADCR.DGON = 1 to update the voltage level.
5. Wait for 500 ns.
6. Set ADCA0DGCTL1.CDG0 = 1 to enable DIAGOUT0.
7. Set ADCA0VCR33.GCTRL[5:0] to ADCAVCR35.GCTRL[5:0] to $000000_B$ to select physical ch0.
8. Set ADCA0VCR33.CNVCLS and ADCA0VCR34.CNVCLS to1 to select normal conversion.
9. Set ADCA0VCR35.CNVCLS = 0 to select hold value conversion.
10. Set ADCA0THACR.SGS[1:0] = $01_B$ to select SG1 to “T&H group A”.
11. Set ADCA0THER.TH0E = 1 to enable T&H circuit ch0.
12. Set ADCA0THGSR.TH0GS = 0 to select T&H circuit ch0 to “T&H group A”.
13. Set ADCA0SGVCSP1.VCSP[5:0] =$100001_B$ to select SG1 start pointer to VCR33.
14. Set ADCA0SGVCEP1.VCEP[5:0] = $100011_B$ to select SG1 end pointer to VCR35.
15. Set ADCA0DGCTL0.PSEL[2:0] = $011_B$ to select $2/3AnV_{REF}$ voltage level.
16. Set ADCA0THSMPSTCR.SMPST = 1 to execute T&H sampling.
17. Wait for 500 ns.
18. Set ADCA0SGSTCR1.SGST = 1 to execute SG1 A/D conversion.
19. Read ADCA0DIR33 to ADCA0DIR35, and check A/D conversion result to see if SG1 A/D conversion has finished.

**NOTES**

To clear ADCAnADCR.DGON, follow the procedure below:

1. Confirm that SGACT of all scan groups is 0 (before scan groups are started) and TRGMD of all scan groups is 0.
2. Use the ADCAnTHER register to disable the diagnosed T&Hk.
3. Clear ADCAnDGCTL0.PSEL[2:0].
4. Clear ADCAnADCR.DGON.

### 31.5.4.2 Diagnosis Mechanism

(1) A reference voltage “A” is applied to one of the reference voltage signals DIAGOUT0 to DIAGOUT2. The T&H circuit holds the voltage “A” and an A/D conversion is performed without using the T&H circuit.

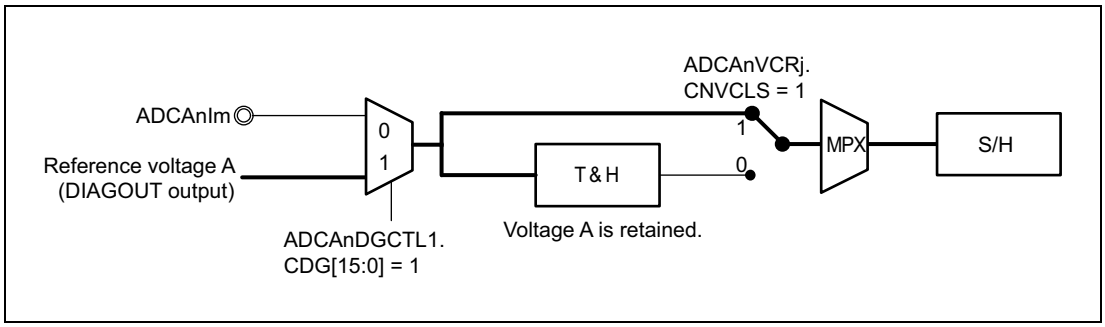



**Figure 31.34 T&H Circuit Diagnostic Mechanisms (1)**

(2) A reference voltage “B” is applied to one of the reference voltage signals DIAGOUT0 to DIAGOUT2. The T&H circuit still holds the voltage “A” and an A/D conversion is performed without using the T&H circuit.
Note that since reference voltage “A” is being held, reference voltage “B” is not held.

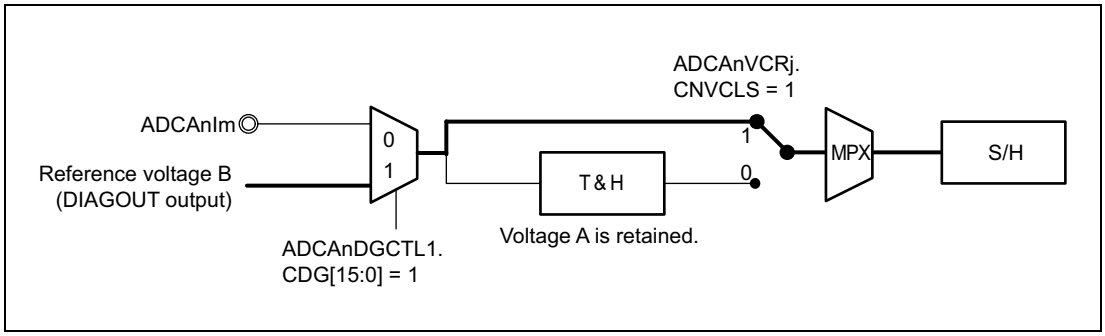



**Figure 31.35 T&H Circuit Diagnostic Mechanisms (2)**

(3) An A/D conversion is performed using the T&H circuit. The T&H circuit continues to hold the voltage A.

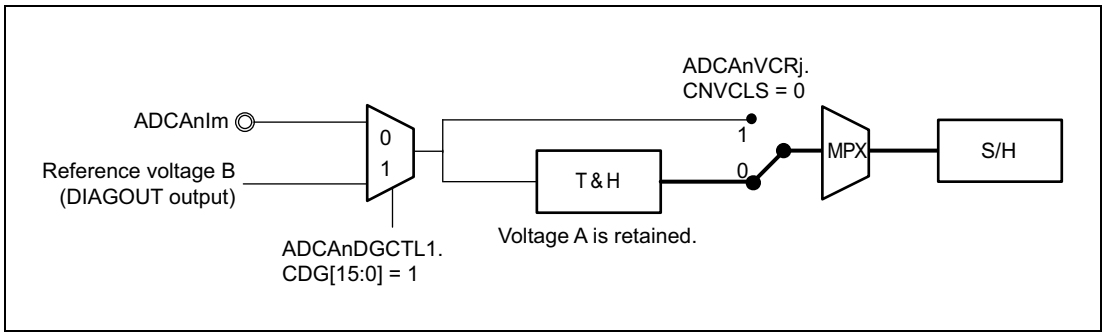



**Figure 31.36 T&H Circuit Diagnostic Mechanisms (3)**

(4) The diagnosis of T&H circuit is successful if the following results are obtained:
1. The first result (step 1) is voltage “A”.
2. The second result (step 2) is voltage “B”.
3. The last result (step 3) is voltage “A” again.

## 31.6 Definition of A/D Conversion Accuracy

A/D conversion accuracy is defined as follows:

- Resolution
  Digital output code value from the A/D converter
- Quantization error
  An error essentially contained in the A/D converter, which is assumed as 1/2 LSB (**Figure 31.37**).
- Offset error
  Deviation of the analog input voltage value from the ideal A/D conversion characteristics when the digital output changes from the minimum voltage value $000_H$ to $001_H$. However, the quantization error is not included.
- Full scale error
  Deviation of the analog input voltage value from the ideal A/D conversion characteristics when the digital output changes from $FFE_H$ to $FFF_H$. However, the quantization error is not included.
- DNL (Differential nonlinear error)
  Deviation between the ideal digital output code width (Vq) and the actual digital output code width (Va), which is assumed as (Va – Vq)/Vq. However, the offset error, the full scale error, and the quantization error are not included.
- INL (Integral nonlinear error)
  Deviation of the actual value from the ideal A/D conversion characteristics between the zero voltage and the full scale voltage, which is assumed as an integral of DNL from $000_H$ to a digital output code. However, the offset error, the full scale error, and the quantization error are not included.
- Absolute accuracy
  Deviation between the digital value and the analog input value. The offset error, the full scale error, the quantization error, DNL, and INL are included.

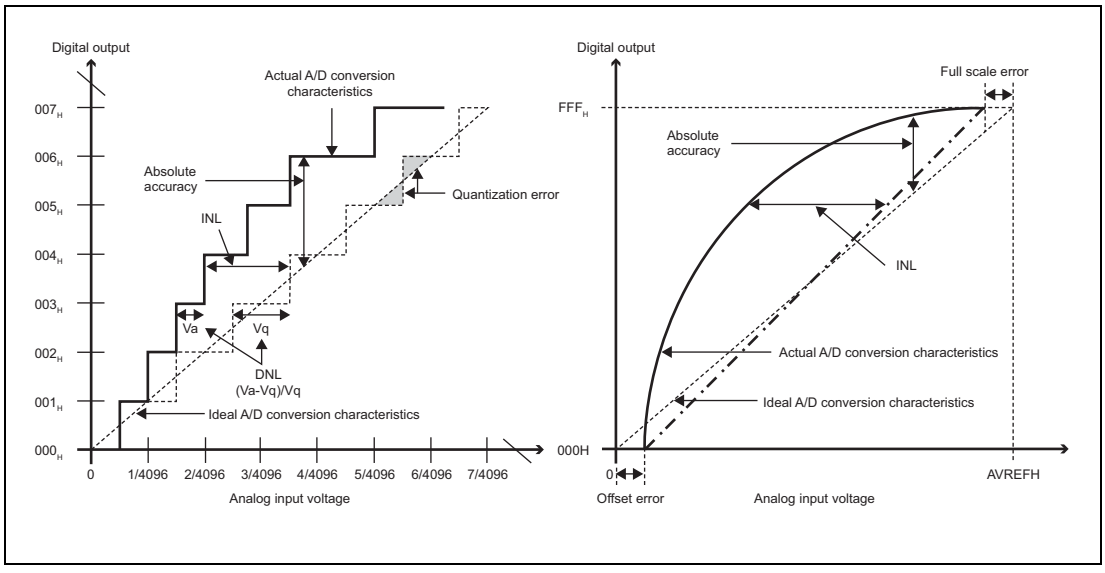



**Figure 31.37 Definition of A/D Conversion Accuracy**

## 31.7 Usage Notes

### 31.7.1 Range of Channel Input Voltage

**CAUTION**

**ADCAnIm and ADCAnImS input voltages should be used within the specification range. If the channel input voltage exceeds AnVREF or falls below AnVSS, the converted accuracy of the channel not guaranteed. When an over-voltage is applied to a pin ADCAnIm and at the same pin the ADC selfdiagnosis (diagnosis of channel multiplexer) is executed an offset voltage to the diagnosis voltage can be measured.**

**When ADCAnDGCTL1 include the pins over-voltage applied, self diagnosis show always over-voltage pin effect. For avoid this issue, ADCAnDGCTL1 should not select over-voltage applied pin.**

### 31.7.2 Notes on Application Design

**(1) Analog input pins (ADCAnIm, ADCAnImS)**

- Ensure that the input voltages on the ADCAnIm and ADCAnImS pins are within the specified ranges. We recommend using diodes with $V_F$ of 0.3V or below to form a clamp to avoid the input of voltages at or above AnVREF and at or below AnVSS. The results of conversion for input voltages at or above AnVREF and at or below AnVSS are undefined and so are not guaranteed. Input of such voltages can also affect the results of conversion on other channels.
- Reduce noise on the analog input pins (ADCAnIm and ADCAnImS) by connecting a resistor Re between the pins and the external sources of analog input signals for conversion and capacitor Ce to the AnVSS pins.
- Avoid analog signal lines crossing digital signal lines and vice versa, since this can introduce noise and reduce performance in A/D conversion.
- We recommend avoiding the driving of large currents through input and output pins near the ADCAnIm and ADCAnImS pins and toggled signals in particular should be kept away from these pins.
- If you are using the standby functions, confirm the stop of A/D conversion. Then, set the ADCAnSGCRx.TRGMD and ADCAnTHER.THkE bits, which are to be effective on standby, to 0.
- If you are using the LPS on ADCA0 (also when standby function is used), set the ADCA0SGCRx.TRGMD bit to 1 and the ADCA0SGTSELx.TxSEL bits to SEQTRG (LPS). For details, see **Section 15, Low-Power Sampler (LPS)**.
- Do not connect a channel to be used with the T&H function to an external analog multiplexer.
- Changes to physical and virtual channels during operation while the T&H function is in use is prohibited.
- Writing to PWM-Diag-related registers while PWM-Diag is not in use is prohibited.

### (2) Power Wiring

The following methods are recommended to minimize the influence of switching noise from digital circuits on A/D converter accuracy.

- Connect markedly thick wiring patterns to the mesh pattern or connect solid patterns to the power-supply lines.
- Insert bypass capacitors between power-supply pins (EVCC and AnVREF) and ground pins (EVSS and AnVSS).
- We recommend separating the analog power supply (AnVREF) from the digital power supplies (EVCC) and providing the voltages from a series regulator. If the analog power supply is to come from the same source as that of the digital power supplies, wire the analog and digital power supplies to an electrolytic capacitor, and provide separate wiring patterns on the board.
  We also recommend inserting a chip inductor in the input for the analog power supply.
  Furthermore, earth the analog and digital grounds to the same point on an electrolytic capacitor, and provide separate wiring patterns for the grounds on the board.
  The analog power supply also serves as the analog reference voltage for this product.

### (3) Variation in A/D converted data

The effects of noise and variations in the power supply voltages lead to dispersal of the results of A/D conversion. Furthermore, noise on the analog input pins (ADCAnIm and ADCAnImS) or on the reference voltage input pins (AnVREF and AnVSS) can lead to the results of A/D conversion being incorrect.

Apply software processing to avoid ill effects on the system of fluctuations in or incorrectness of the results of A/D conversion.

Examples of software handling are described below.

- Use averaged values from several rounds of A/D conversion
- Execute A/D conversion for several time and omit extreme results
- Repeat the processing for abnormalities to check for repeated abnormalities in the case of results of A/D conversion which will cause malfunctions of the system.

There is a possibility that A/D conversion accuracy of high priority SG become worse when following both conditions are applicable.

(1) During the A/D conversion of low priority SG (e.g. SG1), conversion trigger of the high priority SG (e.g. SG3) occur.

(2) The channel T&H function of high priority SG (e.g. SG3) is enabled.

The above case is one of an example of SG combination. The SG priority is SG4 > SG3 > SG2 > SG1. The fluctuation of conversion error depends on the external circuit and devices mounted on the customer board.

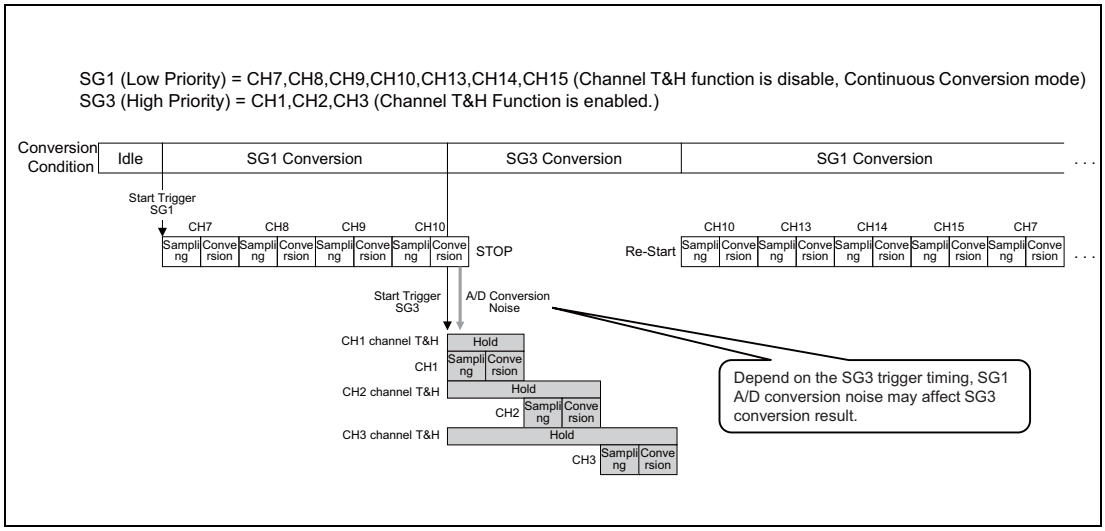



**Figure 31.38 SG priority**

Examples of software handling are described below.

- Low priority SG (e.g. SG1) A/D conversion have to be finished 3 ADCLK before the conversion trigger of high priority SG (e.g. SG3), if the SG3 contains the channels which channel T&H is available.
- If there is a case that high priority SG (e.g. SG3) conversion trigger occur during the conversion of low priority SG (e.g. SG1), disable the high priority SG channel T&H function.
- If the both conditions mentioned in previous page need to be used, adjust high priority SG (e.g. SG3) conversion trigger timing to synchronize with the following timing (the period shown by arrowed line in the following figure) during A/D conversion of low priority SG (e.g. SG1).

Even if trigger timing is adjusted above recommendation time, conversion error specified in Data Sheet cannot be removed.

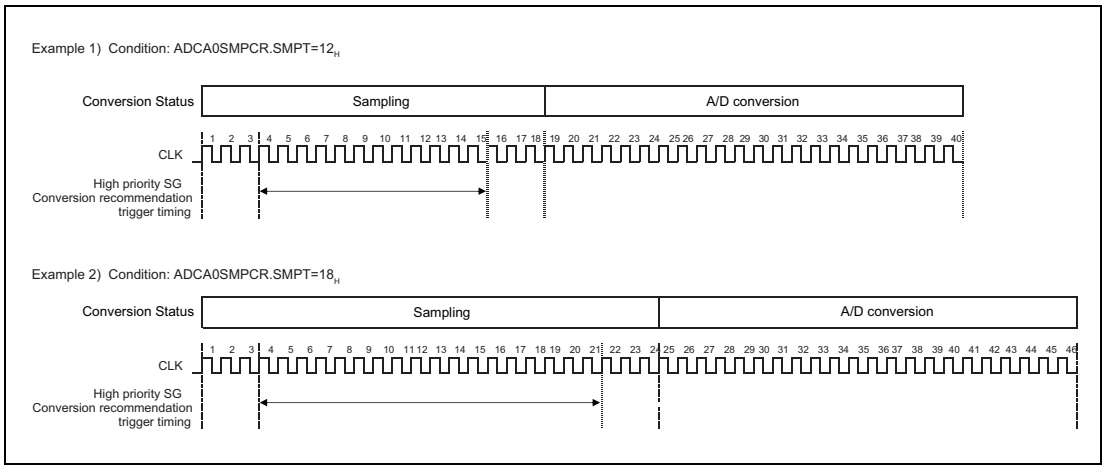



**Figure 31.39 SG conversion trigger timing**

When it is not possible to use the above 3 software handlings, following process is recommended.

- Doing A/D conversion several times and using average of several A/D conversion results.

- Doing continuous A/D conversion several times, remove abnormal conversion result and use only the other results.

- When abnormal A/D conversion result is detected, not to proceed abnormal operation immediately, doing one more A/D conversion before proceeding abnormal operation.

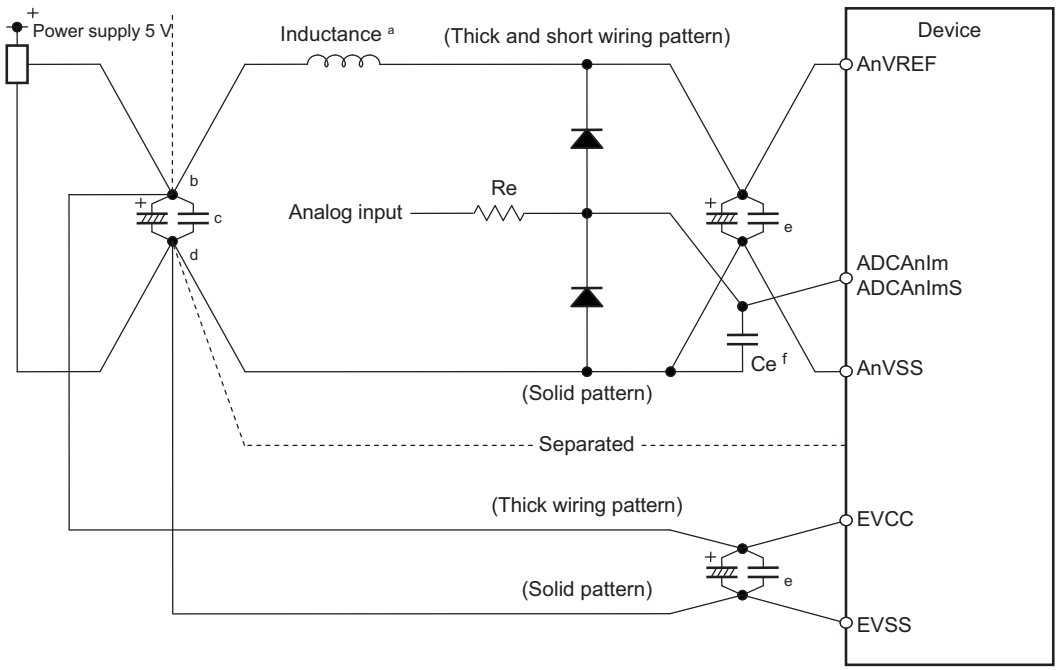
The effect of above process is depend on the external circuit and devices mounted on the customer board. Sufficient evaluation of the system is recommended.

### (4) Alternative Input/Output

Analog input (ADCAnIm, ADCAnImS) pins can be used as port pins.

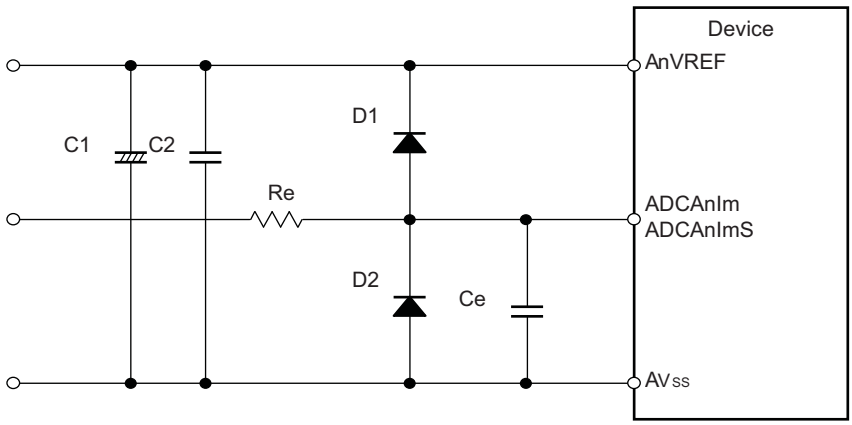
Do not read from input port pins or write to output port pins while an ADCAnIm or ADCAnImS pin function is selected and handling A/D conversion. Doing so may reduce the accuracy of conversion.

Fluctuations in output current from output port pins due to the effects of an external circuit connected to a port pin while A/D conversion is in progress may also reduce the accuracy of conversion. If digital pulses are applied to or digital pulses are output through a pin adjacent to a pin for which A/D conversion is in progress, the A/D converted value may not be as expected due to coupling noise. Accordingly, do not apply pulses to or output pulses from a pin adjacent to a pin for which A/D conversion is in progress.




a) Insert an inductor for the analog power input.
b) Wire the analog and digital supplies to a single point near the 5V power supply and provide separate wiring patterns on the board.
c) Place a bypass capacitor (electrolytic or laminated ceramic) near the common power supply.
d) Wire the analog and digital grounds to a single point near the power supplies, and provide separate wiring patterns for each on the board.
e) Place bypass capacitors (electrolytic or laminated ceramic) near the respective power supply pins.
f) Place capacitor Ce near the analog input pins of the device.

**Figure 31.40 Example of Power Wiring**




**Note:** C1: 4.7uF to 10uF
C2: 0.01uF to 0.1uF
Ce: 5 to 10nF
Re: 0 to 1kΩ
The values above are for reference.
D1, D2: $V_F \leq 0.3$ V

**Figure 31.41 Example of Noise Protection for Analog Input Circuit**

Capacitor C1 is effective for low-frequency noise, and capacitors C2 and Ce are effective for high-frequency noise.

The voltage on an AnVREF pin is undefined immediately after switching from the stopped state to the start of conversion operations, and this may have the effect of reducing the accuracy of conversion. As a countermeasure for this situation, connect capacitors C1 and C2 to the AnVREF pins.

# Section 32 Key Return (KR)

This section contains a generic description of the Key Return (KR) function.

The first part in this section describes the RH850/F1K specific properties, such as the number of units, register base addresses, etc.

The remainder of the section describes the functions and registers of the KR.

## 32.1 Features of RH850/F1K KR

### 32.1.1 Number of Units and Channels

This microcontroller has the following number of KR units.

**Table 32.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of Units | 1 | | |
| Name | KRn (n = 0) | | |

The KR unit has the Key Return function of the following number of channels.

**Table 32.2 KRn Unit Configurations and Channels**

| Unit Name (Channel Name) KRn | No. of Channels | RH850/F1K 100 pins (8 ch) | RH850/F1K 144 pins (8 ch) | RH850/F1K 176 pins (8 ch) |
|---|---|---|---|---|
| KR0 | 8 | √ | √ | √ |

**Table 32.3 Indices**

| Index | Description |
|---|---|
| n | Throughout this section, the individual KR units are identified by the index “n” (n = 0); for example, KRnKRM indicates the key return mode register. |
| m | Throughout this section, the individual KR channels are identified by the index “m” (m = 0 to 7); for example, KRnKRMm indicates the key input enable bit of KRnKRM (key return mode register). |

### 32.1.2 Register Base Address

KRn base addresses are listed in the following table.

KRn register addresses are given as offsets from the base addresses.

**Table 32.4 Register Base Address**

| Base Address Name | Base Address |
|---|---|
| <KR0_base> | FFF7 $8000_H$ |

### 32.1.3 Clock Supply

The KRn clock supply is shown in the following table.

**Table 32.5 Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name | Description |
|---|---|---|---|
| KR0 | PCLK | CPUCLK4 | Module clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CPUCLK4 | |

### 32.1.4 Interrupt Requests

KRn interrupt requests are listed in the following table:

**Table 32.6 Interrupt Requests**

| Unit Interrupt Signal | Description | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **KR0** | | | |
| INTKRn | Key interrupt | 90 | — |

### 32.1.5 Reset Sources

KRn reset sources are listed in the following table. KRn is initialized by these reset sources.

**Table 32.7 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| KR0 | All reset sources (ISORES) |

### 32.1.6 External Input/Output Signals

External input/output signals of KRn are listed below.

**Table 32.8 External Input/Output Signals**

| Unit Signal Name | Description | Alternative Port Pin Signal |
|---|---|---|
| **KR0** | | |
| KRnTPKR7 to KRnTPKR0 | Key input signal | KR0I7 to KR0I0 |

## 32.2 Overview

### 32.2.1 Functional Overview

The Key Return function has the following features:

A key interrupt request signal (INTKRn) can be generated by inputting a falling signal, that goes from high to low, to any of the eight key input pins (KRnTPKR7 to KRnTPKR0).

### 32.2.2 Block Diagram

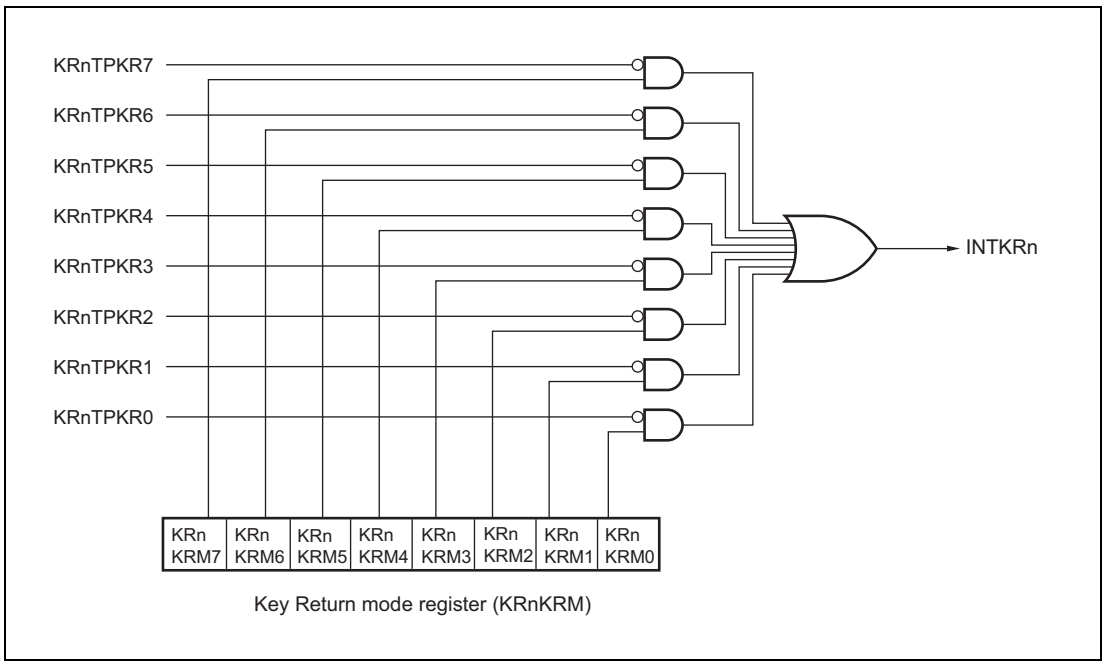



**Figure 32.1 Block Diagram of the Key Return Function**

# 32.3 Registers

## 32.3.1 List of Registers

KR registers are listed in the following table.

For details about <KRn_base>, see **Section 32.1.2, Register Base Address**.

**Table 32.9 List of Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| KRn | Key return mode register | KRnKRM | <KRn_base> |

## 32.3.2 KRnKRM — Key Return Mode Register

This register enables/disables the key input signal detection.

**Access:** This register can be read or written in 8-bit or 1-bit units.

**Address:** <KRn_base>

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | KRnKRM7 | KRnKRM6 | KRnKRM5 | KRnKRM4 | KRnKRM3 | KRnKRM2 | KRnKRM1 | KRnKRM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 32.10 KRnKRM - Key Return Mode Register Contents**

| Bit position | Bit name | Function |
|---|---|---|
| 7 to 0 | KRnKRMm | Enables/disables the key input signal detection.<br>0: Disabled<br>1: Enabled |

## 32.4 Operation

### 32.4.1 Interrupt Request INTKRn

The interrupt request INTKRn is generated when the level of the corresponding key input pin KRnTPKRm is changed from high to low while input to the key input pin KRnTPKRm is enabled (KRnKRM.KRnKRMm = 1).

**Figure 32.2** shows how the interrupt request is generated:

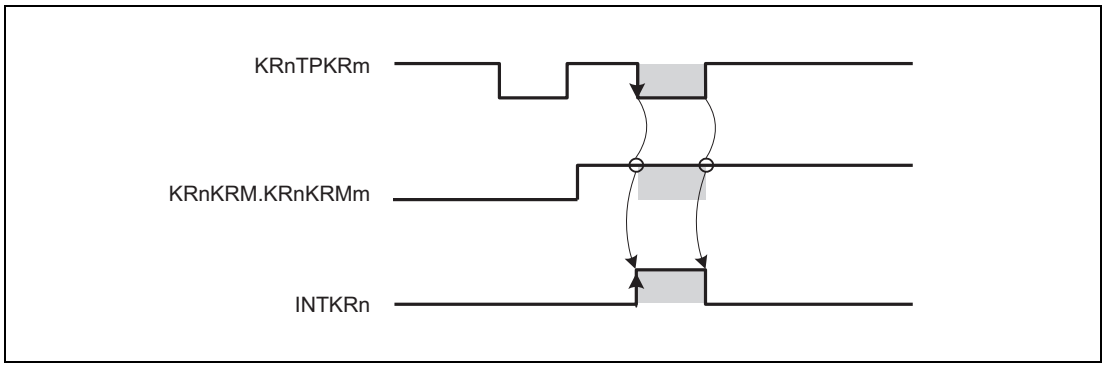



**Figure 32.2 Interrupt Request Generation**

**CAUTIONS**

1. **The change of a key input pin (KRnTPKRm) level from high to low does not trigger another INTKRn if any of the key return input pins are already low. The next INTKRn is only triggered by a key input pin level changing from high to low if all other key input pins are high.**
2. **If the key input value changes at the same time the setting of KRnKRM.KRnKRMm is changed, an unintended key interrupt request INTKRn might be generated. Therefore, mask (disable) INTKRn of the interrupt controller before changing KRnKRM.KRnKRMm from 0 to 1, or from 1 to 0.**

# Section 33 Functional Safety

This section provides an overview of the safety mechanisms included in the RH850/F1K Series.

This microcontroller has been developed as a Safety Element out of Context (SEooC) in accordance with ISO26262.

For more information about the development process and safety mechanisms, please contact our sales office.

The following are the failure detection functions provided by this microcontroller.

## 33.1 Overview

### ECC

Detects failures of memories and data transfer paths and corrects some types of failures.

### Memory Protection

Detects erroneous access to memories and peripheral circuits to protect the data in these elements from erroneous access.

### Clock Monitor

Monitors the clock operation to detect abnormal operations.

For details, see **Section 13, Clock Monitor (CLMA)**.

### Data CRC

Generates CRC to detect data errors.

For details, see **Section 34, Data CRC (DCRA)**.

### Write-Protected Registers

The write-protected registers are protected from inadvertent write access due to erroneous program execution.

For details, see **Section 5, Write-Protected Registers**.

# 33.2 ECC

## 33.2.1 Overview

This product incorporates an ECC for the following memories. The ECC enables detection and correction of errors of the data retained in the memories. The ECC also enables detection and correction of errors produced between the ECC encoder and memories and between memories and ECC decoder.

**Table 33.1 ECC Overview**

| Applicable Memory | | Applicable Data Width [bits] | Operation upon Error Detection | | | | | Failure Insertion |
|---|---|---|---|---|---|---|---|---|
| | | | Detection/ Correction | SYSERR | Interrupt Notice | Error Status | Address Capture | |
| Code flash | | 128 | SEC-DED | DED*2 | SEC-DED | Possible | Possible | Possible |
| Data flash | | 32 | SEC-DED | — | SEC-DED | Possible | Possible | Possible |
| Local RAM (CPU1/Self)<br>Retention RAM (CPU1/Self) | | 32 | SEC-DED | DED*2 | SEC-DED | Possible | Possible | Possible |
| Peripheral RAM*1 | CSIH | 32 | SEC-DED | — | SEC-DED | Possible | Possible | Possible |
| | RSCAN, CAN FD | 32 | SEC-DED | — | SEC-DED | Possible | Possible | Possible |

Note 1. For details of ECC for each peripheral IP, see the corresponding sections below.
- RS-CAN RAM: **Section 21.12, Detection and Correction of Errors in RS-CAN0 RAM** and **Section 21.23, Detection and Correction of Errors in RS-CAN RAM**
- CSIHnRAM: **Section 17.7, Detection and Correction of Errors in CSIHn RAM**

Note 2. For details, see **Section 33.2.2.2, Interrupt Requests** and **Section 33.2.4.2, Interrupt Requests**.

### Applicable Data Width

This is the data width to be ECC encoded.

To write data with a smaller data width than shown, the following processing is required. ECC is also performed for a read in (1).

(1) Reading data to be ECC-encoded including data to be rewritten

(2) Replacing data to be rewritten

(3) Writing back data generated in (2)

### Detection/Correction

SEC-DED: 1-bit errors can be detected and corrected, and 2-bit errors can only be detected.

SED-DED: 1-bit errors and 2-bit errors can only be detected.

### SYSERR

SYSERR can be generated upon error detection.

### Interrupt Notice

An interrupt can be generated upon error detection.

### Error Status

The status of a detected error is retained.

### Address Capture

The address of a detected error is retained.

### Failure Insertion

Self-diagnosis of the ECC decoder error notification function can be performed by using an intentionally generated ECC error.

## 33.2.2 Code Flash ECC

### 33.2.2.1 Overview

RH850/F1K has two code flash ECC decoder circuits implemented, one inside the Processor Element (PE1) and one on the VCI (system interconnect).

**Figure 33.1** shows the location of two ECC decoders for the code flash.

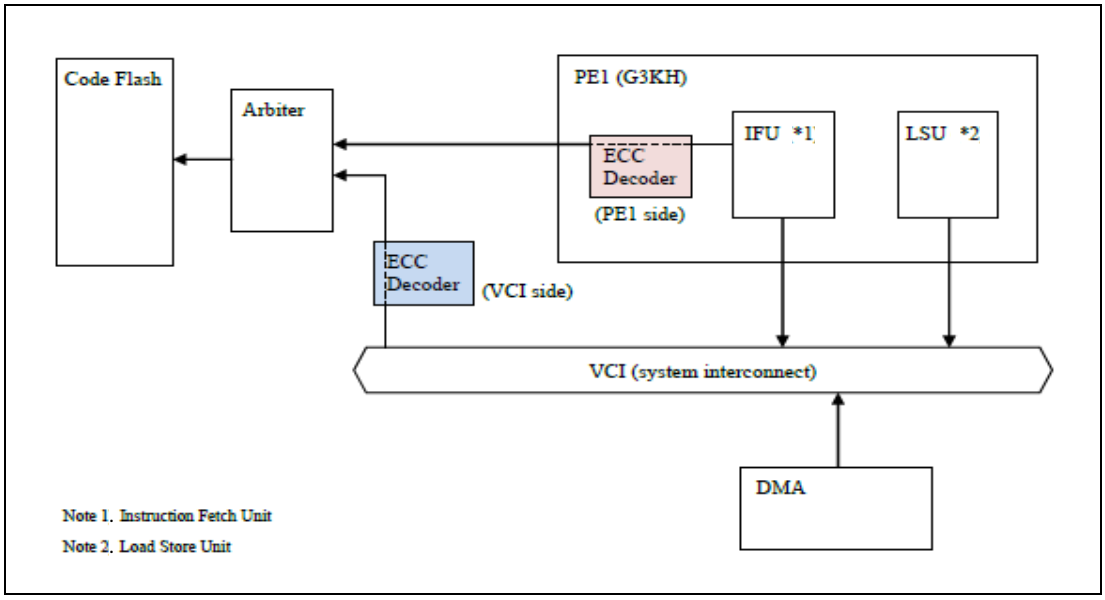



**Figure 33.1 Block Diagram of Code Flash ECC**

The code flash ECC is summarized in the table below.

| Item | Description |
|---|---|
| ECC error detection and correction | ECC error detection and correction can be enabled or disabled.<br>When enabled, either of the following settings can be selected.<br>• 2-bit error detection and 1-bit error detection/correction.<br>• 2-bit error detection / 1-bit error detection.<br>When disabled, neither error detection nor correction is carried out.<br>In the initial state, this function is enabled, and 1-bit errors are detected and corrected, 2-bit errors are detected. |
| Error notification | A notification is sent when an ECC error occurs.<br>**ECC error:**<br>• Enabling or disabling of error (SYSERR exception) notification in the case of detection of ECC 2-bit errors is selectable. For details of the SYSERR, see **Section 3, CPU System**.<br>• Enabling or disabling of error notification in the case of detection of a 1-bit ECC error is selectable.<br>In the initial state, error notification is enabled upon detection of an ECC 2-bit error, and error notification is enabled upon detection of an ECC 1-bit error.<br>However, if an interrupt is masked by the FEINTFMSK register of the interrupt controller, an interrupt processing is not executed. |
| Error status | The detection of ECC 2-bit errors and ECC 1-bit errors can be monitored.<br>The ECC 1-bit error status is set only when no error status has been set.<br>The ECC 2-bit error status is set even when the ECC 1-bit error status is set.<br>A register for clearing the error status is provided. |

| Item | Description |
|---|---|
| Address capture | When no ECC error status has been set, the address at which the first ECC error occurred is captured.<br>In addition, when the retained address source is a 1-bit ECC error, the address of the 2-bit ECC error is also captured. |
| Self-diagnosis | ROM data and the ECC bit can be read directly.<br>Arbitrary data can be written to ROM data and the ECC bit. |
| Inhibiting instruction execution | Generating a SYSERR exception in response to the detection of a 2-bit ECC error during instruction fetching avoids the execution of incorrect instructions. |

#### 33.2.2.2 Interrupt Requests

Interrupt requests for code flash ECC are listed below.

**Table 33.2 Code Flash ECC Interrupt Requests (During CPU Fetching)**

| Unit Interrupt Signal | Description | Name | DMA Trigger Number |
|---|---|---|---|
| — | ECC 1-bit error interrupt of code flash | INTECCSCFLI0 | — |
| — | ECC 2-bit error interrupt of code flash | SYSERR | — |

**Table 33.3 Code Flash ECC Interrupt Requests (During CPU Data Access)**

| Unit Interrupt Signal | Description | Name | DMA Trigger Number |
|---|---|---|---|
| — | ECC 1-bit error interrupt of code flash | INTECCSCFLI0 | — |
| — | ECC 2-bit error interrupt of code flash | INTECCSCFLI0 or SYSERR | — |

**Table 33.4 Code Flash ECC Interrupt Requests (During Bus Master Access except CPU)**

| Unit Interrupt Signal | Description | Name | DMA Trigger Number |
|---|---|---|---|
| — | ECC 1-bit error interrupt of code flash | INTECCSCFLI0 | — |
| — | ECC 2-bit error interrupt of code flash | INTECCSCFLI0 | — |

### 33.2.2.3 List of Registers

Table 33.5 List of Registers

| Module Name | Register Name | Symbol*1 | Address |
|---|---|---|---|
| CFECC_VCI | Code flash ECC control register (VCI) | CFECCCTL_VCI | FFC6 2200$_H$ |
| | Code flash error information control register (VCI) | CFERRINT_VCI | FFC6 2204$_H$ |
| | Code flash status clear register (VCI) | CFSTCLR_VCI | FFC6 2208$_H$ |
| | Code flash error count overflow status register (VCI) | CFOVFSTR_VCI | FFC6 220C$_H$ |
| | Code flash 1st error status register (VCI) | CF1STERSTR_VCI | FFC6 2210$_H$ |
| | Code flash 1st error address register (VCI) | CF1STEADR0_VCI | FFC6 2250$_H$ |
| CFECC_CPU1 | Code flash ECC control register (PE1) | CFECCCTL_PE1 | FFC6 2400$_H$ |
| | Code flash error information control register (PE1) | CFERRINT_PE1 | FFC6 2404$_H$ |
| | Code flash status clear register (PE1) | CFSTCLR_PE1 | FFC6 2408$_H$ |
| | Code flash error count overflow status register (PE1) | CFOVFSTR_PE1 | FFC6 240C$_H$ |
| | Code flash 1st error status register (PE1) | CF1STERSTR_PE1 | FFC6 2410$_H$ |
| | Code flash 1st error address register (PE1) | CF1STEADR0_PE1 | FFC6 2450$_H$ |
| CFECC_VCI | Code flash sub-test control register (VCI) | CFSTSTCTL_VCI | FFC6 2350$_H$ |
| CFECC_CPU1 | Code flash sub-test control register (PE1) | CFSTSTCTL_PE1 | FFC6 2550$_H$ |

Note 1. The registers suffixed with symbols "_VCI" and "_PE1" are provided to ECC controllers corresponding to each access port: registers with "_VCI" are provided to the ECC controller for access from the system interconnect 1 to the code flash and registers with "_PE1" are provided to the ECC controller for access from the CPU1.

### 33.2.2.4 Details of Registers

#### (1) CFECCCTL_VCI/PE1 — Code Flash ECC Control Register

CFECCCTL_VCI/PE1 enables or disables ECC error detection and correction and 1-bit error correction. When writing to CFECCCTL_VCI/PE1, PROT1 and PROT0 need to be $01_B$.

**Access:** CFECCCTL_VCI and CFECCCTL_PE1 can be read or written in 32-bit units.
CFECCCTL_VCIL and CFECCCTL_PE1L can be read or written in 16-bit units.

**Address:** CFECCCTL_VCI: FFC6 $2200_H$
CFECCCTL_VCIL: FFC6 $2200_H$
CFECCCTL_PE1: FFC6 $2400_H$
CFECCCTL_PE1L: FFC6 $2400_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | — | — | SECDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 33.6 CFECCCTL_VCI/PE1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 15 | PROT1 | Enables or disables modification of the ECCDIS and SECDIS bits. The value written is not retained. These bits are always read as 0. Set (PROT1, PROT0) = (0, 1) when writing to CFECCCTL. |
| 14 | PROT0 | |
| 13 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | SECDIS | 1-Bit Error Correction Disable<br>Enables or disables 1-bit error correction when ECC error detection and correction are enabled. Write a value to this bit simultaneously with the setting (PROT1, PROT0) = (0, 1).<br>0: Enables correction of a 1-bit error upon detection.<br>1: Disables correction of a 1-bit error upon detection. |
| 0 | ECCDIS | ECC Disable<br>Enables or disables ECC error detection and correction. Write a value to this bit simultaneously with the setting (PROT1, PROT0) = (0, 1).<br>0: Enables ECC error detection and correction.<br>1: Disables ECC error detection and correction. |

### (2) CFERRINT_VCI/PE1 — Code Flash Error Information Control Register

CFERRINT_VCI/PE1 enables or disables generation of the error notification signal to the interrupt controller upon detection of an ECC 2-bit error or an ECC 1-bit error.

**Access:** CFERRINT_VCI and CFERRINT_PE1 can be read or written in 32-bit units.
CFERRINT_VCIL and CFERRINT_PE1L can be read or written in 16-bit units.
CFERRINT_VCILL and CFERRINT_PE1LL can be read or written in 8-bit units.

**Address:** CFERRINT_VCI: FFC6 2204$_H$
CFERRINT_VCIL: FFC6 2204$_H$
CFERRINT_VCILL: FFC6 2204$_H$
CFERRINT_PE1: FFC6 2404$_H$
CFERRINT_PE1L: FFC6 2404$_H$
CFERRINT_PE1LL: FFC6 2404$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDIE *1 | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 33.7 CFERRINT_VCI/PE1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | DEDIE*1 | ECC 2-Bit Error Notification Enable<br>Enables or disables generation of the error notification signal upon detection of a 2-bit error when ECC error detection and correction are enabled.<br>0: Disables notification of the ECC 2-bit error.<br>1: Enables notification of the ECC 2-bit error. |
| 0 | SEDIE | ECC 1-Bit Error Notification Enable<br>Enables or disables generation of the error notification signal upon detection of a 1-bit error when ECC error detection and correction are enabled.<br>0: Disables notification of the ECC 1-bit error.<br>1: Enables notification of the ECC 1-bit error. |

Note 1. This bit is not supported in CFERRINT_PE1. When writing to this bit in CFERRINT_PE1, always write 1.

Note 2. Regarding the interrupt request, see the **33.2.2.2, Interrupt Requests**.

### (3) CFSTCLR_VCI/PE1 — Code Flash Status Clear Register

CFSTCLR_VCI/PE1 clears the error flags in the error status register (CF1STERSTR_VCI/PE1), the overflow flag in the error overflow status register (CFOVFSTR_VCI/PE1), and the error address register (CF1STEADR0_VCI/PE1).

**Access:** CFSTCLR_VCI and CFSTCLR_PE1 are write-only registers that can be written in 32-bit units.
CFSTCLR_VCIL and CFSTCLR_PE1L are write-only registers that can be written in 16-bit units.
CFSTCLR_VCILL and CFSTCLR_PE1LL are write-only registers that can be written in 8-bit units.

**Address:** CFSTCLR_VCI: FFC6 2208$_H$
CFSTCLR_VCIL: FFC6 2208$_H$
CFSTCLR_VCILL: FFC6 2208$_H$
CFSTCLR_PE1: FFC6 2408$_H$
CFSTCLR_PE1L: FFC6 2408$_H$
CFSTCLR_PE1LL: FFC6 2408$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | STCLR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 33.8 CFSTCLR_VCI/_PE1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | STCLR0 | Error Status Clear<br>0: No effect (Setting 0 does not affect the DEDF0 and SEDF0 flags in CF1STERSTR; ERROVF0 flag in CFOVFSTR; and CF1STEADR0.)<br>1: Writing 1 to this bit clears the DEDF0 and SEDF0 flags in CF1STERSTR; ERROVF0 flag in CFOVFSTR; and CF1STEADR0. |

### (4) CFOVFSTR_VCI/PE1 — Code Flash Error Count Overflow Status Register

CFOVFSTR_VCI/PE1 monitors occurrence of error overflow. If a second error occurs after the first error (= while any of the error flags in the error status register is set), the flag in this register is set. However, if the second error is identical to the first error (both the source and address are same), this flag is not set.

**Access:** CFOVFSTR_VCI and CFOVFSTR_PE1 are read-only registers that can be read in 32-bit units.
CFOVFSTR_VCIL and CFOVFSTR_PE1L are read-only registers that can be read in 16-bit units.
CFOVFSTR_VCILL and CFOVFSTR_PE1LL are read-only registers that can be read in 8-bit units.

**Address:** CFOVFSTR_VCI: FFC6 220C$_H$
CFOVFSTR_VCIL: FFC6 220C$_H$
CFOVFSTR_VCILL: FFC6 220C$_H$
CFOVFSTR_PE1: FFC6 240C$_H$
CFOVFSTR_PE1L: FFC6 240C$_H$
CFOVFSTR_PE1LL: FFC6 240C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ERROVF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 33.9 CFOVFSTR_VCI/PE1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | ERROVF0 | Error Overflow Flag<br>ERROVF0 shows whether a second error occurs while any of the error flags (DEDF0 and SEDF0) in the error status register has occurred, except when both of the error address and source of the second error are the same as those of the first error.<br>0: Did not occur.<br>1: Occurred.<br><br>Clearing condition: Set the STCLR0 bit in CFSTCLR to 1. |

### (5) CF1STERSTR_VCI/PE1 — Code Flash 1st Error Status Register

CF1STERSTR_VCI/PE1 monitors occurrence of the first error. The error status is set if an error occurs while the error flag is 0. If a 2-bit ECC error occurs while the 1-bit ECC error flag is set, the 2-bit ECC error flag is set while retaining the 1-bit ECC error flag.

**Access:** CF1STERSTR_VCI and CF1STERSTR_PE1 are read-only registers that can be read in 32-bit units.
CF1STERSTR_VCIL and CF1STERSTR_PE1L are read-only registers that can be read in 16-bit units.
CF1STERSTR_VCILL and CF1STERSTR_PE1LL are read-only registers that can be read in 8-bit units.

**Address:** CF1STERSTR_VCI: FFC6 2210$_H$
CF1STERSTR_VCIL: FFC6 2210$_H$
CF1STERSTR_VCILL: FFC6 2210$_H$
CF1STERSTR_PE1: FFC6 2410$_H$
CF1STERSTR_PE1L: FFC6 2410$_H$
CF1STERSTR_PE1LL: FFC6 2410$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDF0 | SEDF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 33.10 CF1STERSTR_VCI/PE1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1 | DEDF0 | ECC 2-Bit Error Monitor Flag<br>0: ECC 2-bit error is not detected.<br>1: ECC 2-bit error is detected.<br>Clearing condition: Set the STCLR0 bit in CFSTCLR to 1.<br>Setting condition: ECC 2-bit error is detected when DEDF0 is 0. |
| 0 | SEDF0 | ECC 1-Bit Error Monitor Flag<br>0: ECC 1-bit error is not detected.<br>1: ECC 1-bit error is detected when DEDF0 flag is 0.<br>Clearing condition:<br>Set the STCLR0 bit in CFSTCLR to 1.<br>Setting condition:<br>ECC 1-bit error is detected when DEDF0, SEDF0 are 0. |

### (6) CF1STEADR0_VCI/PE1 — Code Flash 1st Error Address Register

CF1STEADR0_VCI/PE1 holds the address at which an error has occurred.

The error address is updated if an error occurs while all the error flags are 0 in CF1STERSTR_VCI/PE1. The address is updated if an ECC 2-bit error occurs while the ECC 1-bit error flag is set as the first error. If an ECC 2-bit error occurs, the address is not updated.

In addition, the EADR[24:4] bits in this register correspond to the [24:4] bits of the real address. The real address can be calculated by adding the upper address [31:25] bits, to which code flash is mapped, as a base address.

The CF1STERSTR register is cleared by an internal reset, external reset, or by setting the STCLR bit in the CFSTCLR register to 1.

**Access:** CF1STEADR0_VCI and CF1STEADR0_PE1 are read-only registers that can be read in 32-bit units.
CF1STEADR0_VCIL, CF1STEADR0_VCIH, CF1STEADR0_PE1L and CF1STEADR0_PE1H are read-only registers that can be read in 16-bit units.
CF1STEADR0_VCILL, CF1STEADR0_VCILH, CF1STEADR0_VCIHL, CF1STEADR0_VCIHH, CF1STEADR0_PE1LL, CF1STEADR0_PE1LH, CF1STEADR0_PE1HL and CF1STEADR0_PE1HH are read-only registers that can be read in 8-bit units.

**Address:** CF1STEADR0_VCI: FFC6 2250$_H$
CF1STEADR0_VCIL: FFC6 2250$_H$
CF1STEADR0_VCIH: FFC6 2252$_H$
CF1STEADR0_VCILL: FFC6 2250$_H$
CF1STEADR0_VCILH: FFC6 2251$_H$
CF1STEADR0_VCIHL: FFC6 2252$_H$
CF1STEADR0_VCIHH: FFC6 2253$_H$
CF1STEADR0_PE1: FFC6 2450$_H$
CF1STEADR0_PE1L: FFC6 2450$_H$
CF1STEADR0_PE1H: FFC6 2452$_H$
CF1STEADR0_PE1LL: FFC6 2450$_H$
CF1STEADR0_PE1LH: FFC6 2451$_H$
CF1STEADR0_PE1HL: FFC6 2452$_H$
CF1STEADR0_PE1HH: FFC6 2453$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | EADR[24:16] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[15:4] | | | | | | | | | | | | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 33.11 CF1STEADR0_VCI/PE1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 25 | Reserved | When read, the value after reset is returned. |
| 24 to 4 | EADR[24:4] | 1st Error Address<br>Monitors the address of the first error.<br>The error address is updated if an error occurs while all the error flags are 0 in CF1STERSTR. The address is updated if an ECC 2-bit error occurs while the ECC 1-bit error flag is set as the first error. If an ECC 2-bit error occurs, the address is not updated.<br>Clearing condition:<br>Set the STCLR0 bit in CFSTCLR to 1. |
| 3 to 0 | Reserved | When read, the value after reset is returned. |

### (7) CFSTSTCTL_VCI/PE1 — Code Flash Sub-Test Control Register

CFSTSTCTL_VCI/PE1 is used for the ECC test (self-diagnosis). This register is dedicated for code flash. After ECC test mode is enabled by setting ECCTST = 1, the ECC bits can be read directly.

When writing to CFSTSTCTL_VCI/PE1, PROT1 and PROT0 need to be $01_B$.

**Access:** CFSTSTCTL_VCI and CFSTSTCTL_PE1 can be read or written in 32-bit units.
CFSTSTCTL_VCIL and CFSTSTCTL_PE1L can be read or written in 16-bit units.

**Address:** CFSTSTCTL_VCI: FFC6 $2350_H$
CFSTSTCTL_VCIL: FFC6 $2350_H$
CFSTSTCTL_PE1: FFC6 $2550_H$
CFSTSTCTL_PE1L: FFC6 $2550_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | — | — | — | ECCTST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 33.12 CFSTSTCTL_VCI/PE1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 15 | PROT1 | Enables or disables modification of the ECCTST bit.<br>The value written is not retained. These bits are always read as 0.<br>Set (PROT1, PROT0) = (0, 1) when writing to CFSTSTCTL. |
| 14 | PROT0 | |
| 13 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | ECCTST | ECC Test<br>After ECC test mode is enabled by setting ECCTST = 1, ECC bits can be read directly. Write a value to this bit simultaneously with the setting (PROT1, PROT0) = (0, 1). |

Correctly reading instructions from the code flash access port is not possible while ECC test mode is selected (ECCTST = 1). While the access port for the CPU is set to test mode (including during changes to the value of the ECCTST bit), the CPU must run a program from the local RAM or retention RAM and must not fetch instructions from the code flash memory.

The CPU has a small data buffer. If an old value remains in this buffer, the correct value cannot be read even when the ECCTST bit is switched. When switching the ECCTST bit, be sure to clear the data buffer. For how to clear the data buffer, see **Section 3, CPU System**.

From the code flash access port with ECC test mode selected, access must be made by reading 4 bytes aligned to 16n address. The results of reading code flash are as follows:

**Table 33.13 Results of Reading Code Flash**

| Bit Number | Meaning | Bit Position | Description |
|---|---|---|---|
| bit[31:10] | all-0 | 31 to 10 | These bits are always 0. |
| bit[9] | Reserved | 9 | Unknown |
| bit[8:0] | ECC bits | 8 to 0 | ECC bits |

## 33.2.3 Data Flash ECC

### 33.2.3.1 Overview

The data flash ECC is summarized in the table below.

| Item | Description |
|---|---|
| ECC error detection and correction | ECC error detection and correction can be enabled or disabled.<br>When enabled, either of the following settings can be selected.<br>• 2-bit error detection / 1-bit error detection and correction.<br>• 2-bit error detection / 1-bit error detection.<br>When disabled, neither error detection nor correction is carried out.<br>In the initial state, this function is enabled, and 1-bit errors are detected and corrected, 2-bit errors are detected. |
| Error notification | A notification is sent when an ECC error occurs.<br>• It is notified to the interrupt controller. Enabling or disabling of error notification in the case of detection of ECC 1-bit error is selectable.<br>• Enabling or disabling of error notification in the case of detection of ECC 2-bit errors is selectable.<br>In the initial state, error notification is enabled upon detection of an ECC 2-bit error and error notification is disabled upon detection of an ECC 1-bit error. However, if an interrupt is masked by the FEINTFMSK register of the Interrupt Controller, an interrupt processing is not executed. |
| Error status | The detection of ECC 2-bit errors and ECC 1-bit errors is can be monitored.<br>The function is set only while no error status is set.<br>A register for clearing the error status is provided. |
| Address capture | When no ECC error status has been set, the address at which the first ECC error occurred is captured. ECC error address is captured when both SEDF and DEDF of DFERSTR are 0. |
| Self-diagnosis | ROM data and the ECC bit can be read directly.<br>Arbitrary data can be written to ROM data and the ECC bit. |

### 33.2.3.2 Interrupt Requests

The interrupt requests for data flash ECC are shown below.

**Table 33.14 Data Flash ECC Interrupt Requests (During Read Access)**

| Unit Interrupt Signal | Description | Name | DMA Trigger Number |
|---|---|---|---|
| — | ECC 1-bit error interrupt of data flash | INTECCDEEP0 | — |
| — | ECC 2-bit error interrupt of data flash | INTECCDEEP0 | — |

#### 33.2.3.3 List of Registers

Table 33.15 List of Registers

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| DFECC | Data flash ECC control register | DFECCCTL | FFC6 2A00$_H$ |
| | Data flash error status register | DFERSTR | FFC6 2A04$_H$ |
| | Data flash error status clear register | DFERSTC | FFC6 2A08$_H$ |
| | Data flash error overflow status register | DFOVFSTR | FFC6 2A0C$_H$ |
| | Data flash error overflow status clear register | DFOVFSTC | FFC6 2A10$_H$ |
| | Data flash error notification control register | DFERRINT | FFC6 2A14$_H$ |
| | Data flash 1st error address register | DFEADR | FFC6 2A18$_H$ |
| | Data flash test control register | DFTSTCTL | FFC6 2A1C$_H$ |

### 33.2.3.4 Details of Registers

#### (1) DFECCCTL — Data Flash ECC Control Register

DFECCCTL enables or disables ECC error detection and correction and 1-bit error correction. When writing to DFECCCTL, PROT1 and PROT0 need to be $01_B$.

**Access:** DFECCCTL can be read or written in 16-bit units.

**Address:** DFECCCTL: FFC6 $2A00_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | — | — | SEDDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 33.16 DFECCCTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | PROT1 | Enables or disables modification of the ECCDIS and SEDDIS bits. The value written is not retained. These bits are always read as 0. Set (PROT1, PROT0) = (0, 1) when writing to DFECCCTL. |
| 14 | PROT0 | |
| 13 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | SEDDIS | 1-Bit Error Correction Disable<br>Enables or disables 1-bit error correction when ECC error detection and correction are enabled. Write a value to this bit simultaneously with the setting (PROT1, PROT0) = (0, 1).<br>0: Enables correction of a 1-bit error upon detection.<br>1: Disables correction of a 1-bit error upon detection. |
| 0 | ECCDIS | ECC Disable<br>Enables or disables ECC error detection and correction.<br>Write a value to this bit simultaneously with the setting (PROT1, PROT0) = (0, 1).<br>In the initial state, ECC error detection and correction are enabled.<br>0: Enables ECC error detection and correction.<br>1: Disables ECC error detection and correction. |

### (2) DFERSTR — Data Flash Error Status Register

DFERSTR monitors occurrence of errors.

The SEDF bit is set if an ECC 1-bit error is detected while ECC error detection and correction are enabled, and the DEDF bit is set if an ECC 2-bit error is detected.

**Access:** DFERSTR is a read-only register that can be read in 8-bit units.

**Address:** DFERSTR: FFC6 2A04$_H$

**Value after reset:** 00$_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | DEDF | SEDF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 33.17 DFERSTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. |
| 1 | DEDF | ECC 2-Bit Error Monitor Flag<br>0: An ECC 2-bit error is not detected.<br>1: An ECC 2-bit error is detected.<br><br>Clearing condition:<br>Set the ERRCLR bit in DFERSTC to 1.<br>Setting condition:<br>ECC 2-bit error is detected when both SEDF and DEDF are 0. |
| 0 | SEDF | ECC 1-Bit Error Monitor Flag<br>0: An ECC 1-bit error is not detected.<br>1: An ECC 1-bit error is detected.<br><br>Clearing condition:<br>Set the ERRCLR bit in DFERSTC to 1.<br>Setting condition:<br>ECC 1-bit error is detected when both SEDF and DEDF are 0. |

### (3) DFERSTC — Data Flash Error Status Clear Register

DFERSTC clears the error flags in the data flash error status register.

**Access:** DFERSTC is a write-only register that can be written in 8-bit units.

**Address:** DFERSTC: FFC6 2A08$_H$

**Value after reset:** 00$_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | ERRCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 33.18 DFERSTC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | ERRCLR | SEDF/DEDF Flag Clear<br>0: No effect (Setting the ERRCLR bit to 0 does not affect the DEDF and SEDF flags in DFERSTR.)<br>1: The SEDF/DEDF flag in DFERSTR is cleared. |

### (4) DFOVFSTR — Data Flash Error Overflow Status Register

DFOVFSTR monitors occurrence of data flash error overflow.

**Access:** DFOVFSTR is a read-only register that can be read in 8-bit units.

**Address:** DFOVFSTR: FFC6 2A0C$_H$

**Value after reset:** 00$_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | ERROVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 33.19 DFOVFSTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | ERROVF | Error Overflow Flag<br>ERROVF is set if an ECC error occurs while the error address register is full.<br>0: Did not occur.<br>1: Occurred.<br>Clearing condition:<br>Set the ERROVFCLR bit in data flash error overflow status clear register. |

### (5) DFOVFSTC — Data Flash Error Overflow Status Clear Register

DFOVFSTC clears the data flash error overflow flag.

**Access:** DFOVFSTC is a write-only register that can be written in 8-bit units.

**Address:** DFOVFSTC: FFC6 2A10$_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | ERROVFCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 33.20 DFOVFSTC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | ERROVFCLR | Error Overflow Flag Clear<br>0: No effect (Setting the ERROVFCLR bit to 0 does not affect the ERROVF flag in DFOVFSTR.)<br>1: The ERROVF flag in DFOVFSTR is cleared. |

### (6) DFERRINT — Data Flash Error Notification Control Register

DFERRINT enables or disables generation of the error notification signal upon detection of an ECC 2-bit error or an ECC 1-bit error.

**Access:** DFERRINT can be read or written in 8-bit units.

**Address:** DFERRINT: FFC6 2A14$_H$

**Value after reset:** $02_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 33.21 DFERRINT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | DEDIE | ECC 2-Bit Error Notification Control<br>Enables or disables generation of the error notification signal upon detection of a 2-bit error when ECC error detection and correction are enabled.<br>0: Disables notification of the ECC 2-bit error.<br>1: Enables notification of the ECC 2-bit error. |
| 0 | SEDIE | ECC 1-Bit Error Notification Control<br>Enables or disables generation of the error notification signal upon detection of a 1-bit error when ECC error detection and correction are enabled.<br>0: Disables notification of the ECC 1-bit error.<br>1: Enables notification of the ECC 1-bit error. |

### (7) DFEADR — Data Flash 1st Error Address Register

DFEADR holds the address at which an ECC error has occurred while both of the SEDF and DEDF bits in the data flash error status register are 0.

**Access:** DFEADR is a read-only register that can be read in 32-bit units.

**Address:** DFEADR: FFC6 2A18$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | DFEADR[20:16] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DFEADR[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 33.22 DFEADR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is returned. |
| 20 to 2 | DFEADR[20:2] | ECC Error Address<br>DFEADR is read-only field to monitor the address at which an ECC error has occurred.<br>This register holds an internal address.<br>Convert it to the actual address by adding the data flash base address FF20 0000$_H$. |
| 1 to 0 | Reserved | When read, the value after reset is returned. |

### (8) DFTSTCTL — Data Flash Test Control Register

DFTSTCTL is used for the ECC test.

After ECC test mode is enabled by setting ECCTST = 1, the ECC bits can be read.

When writing to DFTSTCTL, PROT1 and PROT0 need to be $01_B$.

**Access:** DFTSTCTL can be read or written in 16-bit units.

**Address:** FFC6 2A1C$_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | — | — | — | ECCTST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 33.23 DFTSTCTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | PROT1 | Enables or disables modification of the ECCTST bit.<br>The value written is not retained. These bits are always read as 0.<br>Set (PROT1, PROT0) = (0, 1) when writing to DFTSTCTL. |
| 14 | PROT0 | |
| 13 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | ECCTST | ECC Test<br>Sets ECC test mode.<br>Write a value to this bit simultaneously with the setting (PROT1, PROT0) = (0, 1). |

## 33.2.4 Local RAM (Including the Retention RAM) ECC

### 33.2.4.1 Overview

**CAUTION**

**The retention RAM is a part of the local RAM. The ECC for the retention RAM is shared with the local RAM. Therefore, use for the retention RAM the same register as that for the local RAM.**

Local RAM ECC of CPU1 is summarized in the table below.

| Item | Description |
|---|---|
| ECC error detection and correction | ECC error detection and correction can be enabled or disabled.<br>When enabled, either of the following settings can be selected.<br>• 2-bit error detection and 1-bit error detection/correction.<br>• 2-bit error detection and 1-bit error detection.<br>When disabled, neither error detection nor correction is carried out.<br>In the initial state, the ECC function is enabled, and 1-bit errors are detected and corrected, 2-bit errors detected. |
| Error notification | A notification is sent when an ECC error occurs.<br>**ECC error:**<br>• Enabling or disabling of error (SYSERR exception) notification in the case of detection of ECC 2-bit errors is selectable. For details of the SYSERR, see **Section 3, CPU System**.<br>• Enabling or disabling of error notification in the case of detection of ECC 1-bit errors is selectable.<br>In the initial state, notification of the 2-bit error is enabled and notification of the 1-bit error is enabled. However, if an interrupt is masked by the FEINTFMSK register of the interrupt controller, an interrupt processing is not executed. |
| Error status | The detection of ECC 2-bit errors and ECC 1-bit errors is can be monitored.<br>The function is set only while no error status is set.<br>A register for clearing the error status is provided. |
| Address capture | When no error status has been set, the address at which the first error occurred is captured.<br>In addition, when the retained address source is a 1-bit ECC error, the address of the 2-bit ECC error is also captured. |
| Self-diagnosis | Arbitrary data can be written to RAM data and the ECC bit.<br>RAM data and the ECC bit can be read directly. |
| Others | Generating a SYSERR exception in response to the detection of a 2-bit ECC error during instruction fetching prevents the execution of incorrect instructions. |

**CAUTION**

**When ECC error detection/correction for the local RAM is enabled for access, initialize the RAM with the 32-bit length of RAM access before the RAM is used. If the RAM before initialization is read, an FE-level maskable interrupt or SYSERR exceptional processing may be generated.**

**Moreover, if the RAM is not initialized with the 32-bit length (for example, initialized with 8- or 16-bit length of access), an FE-level maskable interrupt or SYSERR exceptional processing may be generated.**

#### 33.2.4.2 Interrupt Requests

Local RAM ECC interrupt requests are listed below.

**Table 33.24 Local RAM ECC Interrupt Requests (During CPU Access)**

| Unit Interrupt Signal | Description | Name | DMA Trigger Number |
|---|---|---|---|
| — | ECC 1-bit error interrupt of RAM | INTECCRAM | — |
| — | ECC 2-bit error interrupt of RAM | INTECCRAM or SYSERR | — |

**Table 33.25 Local RAM ECC Interrupt Requests (During Read Access except CPU Access)**

| Unit Interrupt Signal | Description | Name | DMA Trigger Number |
|---|---|---|---|
| — | ECC 1-bit error interrupt of RAM | INTECCRAM | — |
| — | ECC 2-bit error interrupt of RAM | INTECCRAM | — |

### 33.2.4.3 List of Registers

Table 33.26 List of Registers

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| LRTST | Local RAM test control register (PE1) | LRTSTCTL_PE1 | FFC6 $5004_H$ |
| | Local RAM test data read buffer 0 (PE1) | LRTDATBF0_PE1 | FFC6 $5008_H$ |
| LRECC | Local RAM ECC control register (PE1) | LRECCCTL_PE1 | FFC6 $5400_H$ |
| | Local RAM error information control register (PE1) | LRERRINT_PE1 | FFC6 $5404_H$ |
| | Local RAM status clear register (PE1) | LRSTCLR_PE1 | FFC6 $5408_H$ |
| | Local RAM error count overflow status register (PE1) | LROVFSTR_PE1 | FFC6 $540C_H$ |
| | Local RAM 1st error status register (PE1) | LR1STERSTR_PE1 | FFC6 $5410_H$ |
| | Local RAM 1st error address register 0 (PE1) | LR1STEADR0_PE1 | FFC6 $5450_H$ |

### 33.2.4.4 Details of Registers

#### (1) LRTSTCTL_PE1 — Local RAM Test Control Register

LRTSTCTL is used for the ECC test (self-diagnosis). After ECC test mode is enabled by setting ECCTST = 1, any data can be written to the ECC bits. The DATSEL bit is used to select RAM data or the ECC bits.

When writing to LRTSTCTL, PROT1 and PROT0 need to be $01_B$.

**Access:** LRTSTCTL_PE1 register can be read or written in 32-bit units.
LRTSTCTL_PE1L register can be read or written in 16-bit units.

**Address:** LRTSTCTL_PE1: FFC6 $5004_H$
LRTSTCTL_PE1L: FFC6 $5004_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | — | — | ECCTST | DATSEL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 33.27 LRTSTCTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 15 | PROT1 | Enables or disables modification of the ECCTST and DATSEL bits.<br>The value written is not retained. These bits are always read as 0.<br>Set (PROT1, PROT0) = (0, 1) when writing to LRTSTCTL. |
| 14 | PROT0 | |
| 13 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | ECCTST | ECC Test<br>After ECC test mode is enabled by setting ECCTST = 1, the ECC bits can be read directly. Write a value to this bit simultaneously with the setting (PROT1, PROT0) = (0, 1). |
| 0 | DATSEL | Data Select<br>This bit is valid when ECCTST = 1. This bit selects the RAM bit which can be accessed when writing. Write a value to this bit simultaneously with the setting (PROT1, PROT0) = (0, 1).<br>0: RAM data is selected.<br>1: The ECC bits are selected. |

**CAUTION**

When ECC test mode for the local RAM is enabled (ECCTST = 1), the local RAM should be accessed in 4-byte units.

### (2) LRTDATBF0_PE1 — Local RAM Test Data Read Buffer 0

In ECC test mode (self-diagnosis), the ECC bits can be read. If the local RAM is read while ECCTST = 1 in the local RAM test control register LRTSTCTL, reading from the local RAM reads out the ECC bits, and these bits are stored in this buffer.

**Access:** LRTDATBF0_PE1 register is a read only register that can be read in 32-bit units.

**Address:** FFC6 $5008_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | LRDATABF[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 33.28 LRTDATBF0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | When read, the value after reset is returned. |
| 6 to 0 | LRDATABF [6:0] | These bits are valid when ECCTST = 1 (selecting test mode) in the local RAM test control register.<br>When reading from the local RAM, the ECC bits for the local RAM are stored in LRTDATABF[6:0]. |

### (3) LRECCCTL_PE1 — Local RAM ECC Control Register

LRECCCTL enables or disables ECC error detection and correction and 1-bit error correction.

When writing to LRECCCTL, PROT1 and PROT0 need to be $01_B$.

**Access:** LRECCCTL_PE1 register can be read or written in 32-bit units.
LRECCCTL_PE1L register can be read or written in 16-bit units.

**Address:** LRECCCTL_PE1: FFC6 $5400_H$
LRECCCTL_PE1L: FFC6 $5400_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | — | — | SECDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 33.29 LRECCCTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 15 | PROT1 | Enables or disables modification of the ECCDIS and SECDIS bits. The value written is not retained. These bits are always read as 0. Set (PROT1, PROT0) = (0, 1) when writing to LRECCCTL. |
| 14 | PROT0 | |
| 13 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | SECDIS | 1-Bit Error Correction Disable<br>Enables or disables 1-bit error correction when ECC error detection and correction are enabled. Write a value to this bit simultaneously with the setting (PROT1, PROT0) = (0, 1).<br>0: Enables correction of a 1-bit error upon detection.<br>1: Disables correction of a 1-bit error upon detection. |
| 0 | ECCDIS | ECC Disable<br>Enables or disables ECC error detection and correction. Write a value to this bit simultaneously with the setting (PROT1, PROT0) = (0, 1).<br>0: Enables ECC error detection and correction.<br>1: Disables ECC error detection and correction. |

### (4) LRERRINT_PE1 — Local RAM Error Information Control

LRERRINT enables or disables generation of the error notification signal to the interrupt controller upon detection of an ECC 2-bit error or an ECC 1-bit error.

**Access:** LRERRINT_PE1 register can be read or written in 32-bit units.
LRERRINT_PE1L register can be read or written in 16-bit units.
LRERRINT_PE1LL register can be read or written in 8-bit units.

**Address:** LRERRINT_PE1: FFC6 5404$_H$
LRERRINT_PE1L: FFC6 5404$_H$
LRERRINT_PE1LL: FFC6 5404$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 33.30 LRERRINT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | DEDIE | ECC 2-Bit Error Notification Enable<br>Enables or disables generation of the error notification signal upon detection of a 2-bit error when ECC error detection and correction are enabled.<br>0: Disables notification of the ECC 2-bit error.<br>1: Enables notification of the ECC 2-bit error. |
| 0 | SEDIE | ECC 1-Bit Error Notification Enable<br>Enables or disables generation of the error notification signal upon detection of a 1-bit error when ECC error detection and correction are enabled.<br>0: Disables notification of the ECC 1-bit error.<br>1: Enables notification of the ECC 1-bit error. |

### (5) LRSTCLR_PE1 — Local RAM Status Clear Register

LRSTCLR clears the error flags in the error status register (LR1STERSTR), the overflow flag in the error overflow status register (LROVFSTR), and the error address register (LR1STEADR0). LRSTCLR is a write-only register and is always read as 0.

**Access:** LRSTCLR_PE1 register is a write only register that can be written in 32-bit units.
LRSTCLR_PE1L register is a write only register that can be written in 16-bit units.
LRSTCLR_PE1LL register is a write only register that can be written in 8-bit units.

**Address:** LRSTCLR_PE1: FFC6 5408$_H$
LRSTCLR_PE1L: FFC6 5408$_H$
LRSTCLR_PE1LL: FFC6 5408$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | STCLR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 33.31 LRSTCLR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | STCLR0 | Error Status Flag Clear<br>Writing 1 to this bit clears the DEDF0 and SEDF0 flags in LR1STERSTR; ERROVF0 flag in LROVFSTR; and LR1STEADR0. |

### (6) LROVFSTR_PE1 — Local RAM Error Count Overflow Status Register

LROVFSTR monitors occurrence of error overflow. If a second error occurs after the first error (= while any of the error flags in the error status register is set), the flag in this register is set. However, if the second error is identical to the first error (both the source and address are same), this flag is not set. ERROVF0 is cleared by an internal reset, an external reset, or setting the STCLR0 bit to 1 in LRSTCLR.

**Access:** LROVFSTR_PE1 register is a read only register that can be read in 32-bit units.
LROVFSTR_PE1L register is a read only register that can be read in 16-bit units.
LROVFSTR_PE1LL register is a read only register that can be read in 8-bit units.

**Address:** LROVFSTR_PE1: FFC6 540C$_H$
LROVFSTR_PE1L: FFC6 540C$_H$
LROVFSTR_PE1LL: FFC6 540C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ERROVF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 33.32 LROVFSTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | ERROVF0 | Error Overflow Flag<br>ERROVF0 is set if a second error occurs while any of the error flags (DEDF0 and SEDF0) in the error status register is set, except when both of the error address and source of the second error are the same as those of the first error. |

### (7) LR1STERSTR_PE1 — Local RAM 1st Error Status Register

LR1STERSTR monitors occurrence of the first error. The error status is set if an error occurs while the error flag is 0.

If more than one error occurs simultaneously, all the corresponding error flags are set. LR1STERSTR is cleared by an internal reset, an external reset, or setting 1 to the STCLR0 bit in LRSTCLR.

**Access:** LR1STERSTR_PE1 register is a read only register that can be read in 32-bit units.
LR1STERSTR_PE1L register is a read only register that can be read in 16-bit units.
LR1STERSTR_PE1LL register is a read only register that can be read in 8-bit units.

**Address:** LR1STERSTR_PE1: FFC6 5410$_H$
LR1STERSTR_PE1L: FFC6 5410$_H$
LR1STERSTR_PE1LL: FFC6 5410$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDF0 | SEDF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 33.33 LR1STERSTR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1 | DEDF0 | ECC 2-Bit Error Monitor Flag<br>0: ECC 2-bit error is not detected.<br>1: ECC 2-bit error is detected.<br><br>Clearing condition:<br>Set the STCLR0 bit in LRSTCLR to 1.<br>Setting condition:<br>ECC 2-bit error is detected with DEDF0 being 0. |
| 0 | SEDF0 | ECC 1-Bit Error Monitor Flag<br>0: ECC 1-bit error is not detected.<br>1: ECC 1-bit error is detected.<br><br>Clearing condition:<br>Set the STCLR0 bit in LRSTCLR to 1.<br>Setting condition:<br>ECC 1-bit error is detected with both SEDF0 and DEDF0 being 0. |

### (8) LR1STEADR0_PE1 — Local RAM 1st Error Address Register 0

LR1STEADR0 holds the address at which an error has occurred.

The error address is set if an error occurs while error flags are 0 in LR1STERSTR. The address is updated if an ECC 2-bit error occurs while the ECC 1-bit error flag is set as the first error. Once an ECC 2-bit error occurs, the address is not updated.

Since this register holds the internal address, add the base address of the associated memory to transform the internal address to the real address. LR1STEADR0 is cleared by an internal reset, an external reset, or setting the STCLR0 bit to 1 in LRSTCLR.

**Access:** LR1STEADR0_PE1 register is a read only register that can be read in 32-bit units.
LR1STEADR0_PE1L and LR1STEADR0_PE1H registers are read only registers that can be read in 16-bit units.
LR1STEADR0_PE1LL, LR1STEADR0_PE1LH, and LR1STEADR0_PE1HL registers are read only registers that can be read in 8-bit units.

**Address:** LR1STEADR0_PE1: FFC6 5450$_H$
LR1STEADR0_PE1L: FFC6 5450$_H$,
LR1STEADR0_PE1H: FFC6 5452$_H$
LR1STEADR0_PE1LL: FFC6 5450$_H$,
LR1STEADR0_PE1LH: FFC6 5451$_H$,
LR1STEADR0_PE1HL: FFC6 5452$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | EADR[18:16] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 33.34 LR1STEADR0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 19 | Reserved | When read, the value after reset is returned. |
| 18 to 2 | EADR[18:2] | 1st Error Address<br>Monitors the address of the first error.<br>The error address is held if an error occurs while all the error flags are 0 in LR1STERSTR. The address is updated if an ECC 2-bit error occurs while the ECC 1-bit error flag is set as the first error. Once an ECC 2-bit error occurs, the address is not updated. |
| 1 to 0 | Reserved | When read, the value after reset is returned. |

## 33.2.5 Peripheral RAM ECC

CSIH and RS-CAN have the ECC function for RAM. For details of these ECC function, see **Section 17.7, Detection and Correction of Errors in CSIHn RAM**, **Section 21.12, Detection and Correction of Errors in RS-CAN0 RAM** and **Section 21.23, Detection and Correction of Errors in RS-CAN RAM**.

# 33.3 Memory Protection

## 33.3.1 Overview

This product incorporates the memory protection function to prevent erroneous accesses to data in memories and control registers of the peripheral circuits.

- MPU
  The CPU protects memory against illegal access by itself. The CPU does not generate the signals for access to addresses where access is prohibited by the MPU. For details, see the *RH850 Family User's Manual: Software*.
- Slave Guard
  A specific memory is protected against illegal accesses from any bus master. Slave guard includes the following guard types. The details of each type are given in the following sections.
  - PEG
    The local RAM is protected against illegal accesses. However, accesses from the CPU incorporating the local RAM itself are excluded. For example, accesses from the CPU1 to local RAM in the CPU1 are not rejected by the PEG. For details, see **Section 3, CPU System**.
  - PBG
    The control registers in the peripheral circuits are protected against illegal accesses. For details, see **Section 33.3.2, PBG**.
  - PBGC
    The CPU system has a dedicated PBG function which is called a PBG for CPU system. For details, see **Section 33.3.3, PBG for CPU System**.

#### 33.3.1.1 Identifiers for Slave Guard

For the slave guard function, the type of illegal accesses to be rejected can be designated using the following identifiers.

**Table 33.35 Identifiers for Slave Guard**

| Identifier | Function |
|---|---|
| UM | When the CPU makes an access, this indicates the operating mode of the CPU.<br>0: Supervisor mode<br>1: User mode<br>When the DMAC makes an access, the value of this identifier is the value in the channel master setting register.<br>When another master makes an access, the value of this identifier is always 0. |
| SPID | When the CPU makes an access, this indicates the system protection identifier SPID that is assigned to the CPU.<br>When the DMAC makes an access, the value of this identifier is the value in the channel master setting register.<br>When another master makes an access, the value of this identifier is always $00_B$. |
| PEID | This indicates the access source bus master.<br>$000_B$: Reserved<br>$001_B$: CPU1<br>$010_B$: Reserved<br>$011_B$: Reserved<br>$100_B$: Reserved<br>$101_B$: Reserved<br>$110_B$: Reserved<br>$111_B$: Reserved<br>When the DMAC makes an access, the value of this identifier is the value in the channel master setting register. |

### 33.3.2 PBG

The PBG module is divided into several PBG groups, each of which is provided with a maximum of 16 protection channels. A single PBG channel can designate the access against which a single peripheral circuit should be protected.

Each PBG group can hold the information of the access that has been rejected.

The following table lists the peripheral circuits to be protected, the corresponding PBG group names, and the PBG channel numbers.

**Table 33.36 PBG Groups and Channels and Target Modules (1/4)**

| PBG Group | Group No. | PBG Channel Number | Protection Target Module | Target Register |
|---|---|---|---|---|
| PBG10 | 00 | 0 | FENMI | WDTNMIF<br>WDTNMIFC |
| | | 1 | FEINT | FEINTF<br>FEINTFMSK<br>FEINTFC |
| | | 2 | INTC1/INTC2 | SELB_INTC1<br>SELB_INTC2 |
| | | 3 | ADCA0 | All registers in ADCA0 |
| | | 4 | KR0 | KR0KRM |
| | | 5 | PORT_ISO (Group A) | Registers of P9, P10, P11, P12, P18, P20, and AP1 (except Pn, PSRn, PPRn, PNOTn, and PIBCn) |
| | | 6 | PORT_AWO (Group A) | Registers of P0, P1, P2, P8, AP0, and IP0 (except Pn, PSRn, PPRn, PNOTn, and PIBCn) |
| | | 7 | PORT_JTAG (Group A) | JP0 register (except JP0, JPSR0, JPPR0, JPNOT0, and JPIBC0) |
| | | 8 | RLIN30 | All registers in RLIN30 |
| | | 9 | RLIN31 | All registers in RLIN31 |
| | | 10 | RLIN32 | All registers in RLIN32 |
| | | 11 | RLIN33 | All registers in RLIN33 |
| | | 12 | RLIN34 | All registers in RLIN34 |
| | | 13 | RLIN35 | All registers in RLIN35 |
| PBG11 | 01 | 0 | DNFA_TAUD0 | DNFATAUD0ICTL<br>DNFATAUD0IEN |
| | | 1 | DNFA_ADCTL0 | DNFAADCTL0CTL<br>DNFAADCTL0EN |
| | | 2 | FCLA_ADC0 | FCLA0CTLn_ADC0 (n = 0 to 2) |
| | | 3 | FCLA_NMI | FCLA0CTL0_NMI |
| | | 4 | FCLA_INTPL | FCLA0CTLn_INTPL (n = 0 to 7) |
| | | 5 | FCLA_INTPH | FCLA0CTLn_INTPH (n = 0 to 7) |
| | | 6 | DNFA_ENCA0 | DNFAENCA0ICTL<br>DNFAENCA0IEN |
| | | 7 | DNFA_ADCTL1 | DNFAADCTL1CTL<br>DNFAADCTL1EN |
| | | 8 | FCLA_ADC1 | FCLA0CTLn_ADC1 (n=0 to 2) |
| | | 9 | DNFA_TAUB0I | DNFATAUB0ICTL<br>DNFATAUB0IEN |

Table 33.36 PBG Groups and Channels and Target Modules (2/4)

| PBG Group | Group No. | PBG Channel Number | Protection Target Module | Target Register |
|---|---|---|---|---|
| PBG11 | 01 | 10 | DNFA_TAUB1I | DNFATAUB1ICTL<br>DNFATAUB1IEN |
| | | 11 | Reserved area | — |
| | | 12 | PORT_ISO (Group B) | Registers of P9, P10, P11, P12, P18, P20, and AP1 (Pn, PSRn, PPRn, PNOTn, and PIBCn) |
| | | 13 | PORT_AWO (Group B) | Registers of P0, P1, P2, P8, AP0, and IP0 (Pn, PSRn, PPRn, PNOTn, and PIBCn) |
| | | 14 | PORT_JTAG (Group B) | JP0 register (JP0, JPSR0, JPPR0, JPNOT0, and JPIBC0) |
| PBG12 | 07 | 0 | RLIN2_Global0 | All global registers in RLN240 |
| | | 1 | RLIN20 | All channel registers in RLN2400 |
| | | 2 | RLIN21 | All channel registers in RLN2401 |
| | | 3 | RLIN22 | All channel registers in RLN2402 |
| | | 4 | RLIN23 | All channel registers in RLN2403 |
| | | 5 | RLIN2_Global1 | All global registers in RLN241 |
| | | 6 | RLIN24 | All channel registers in RLN2414 |
| | | 7 | RLIN25 | All channel registers in RLN2415 |
| | | 8 | RLIN26 | All channel registers in RLN2416 |
| | | 9 | RLIN27 | All channel registers in RLN2417 |
| | | 10 | RLIN2_Global2 | All global registers in RLN242 |
| | | 11 | RLIN28 | All channel registers in RLN2428 |
| | | 12 | RLIN29 | All channel registers in RLN2429 |
| PBG13 | 08 | 0 | DCRA0 | All registers in DCRA0 |
| | | 1 | DCRA1 | All registers in DCRA1 |
| | | 2 | DCRA2 | All registers in DCRA2 |
| | | 3 | DCRA3 | All registers in DCRA3 |
| | | 4 | $I^2C$ | All registers in $I^2C$ |
| | | 5 | ECC test | SELB_READTEST |
| PBG20 | 02 | 0 | TAUD0 (Group A) | All registers in TAUD0 (except SELB_TAUD0I) |
| | | 1 | TAUD0 (Group B) | SELB_TAUD0I |
| | | 2 | TAUJ0 (Group A) | All registers in TAUJ0 (except SELB_TAUJ0I) |
| | | 3 | TAUJ0 (Group B) | SELB_TAUJ0I |
| | | 4 | RTCA0 | All registers in RTCA0 |
| | | 5 | WDTA0 | All registers in WDTA0 |
| | | 6 | WDTA1 | All registers in WDTA1 |
| | | 7 | Reserved area | — |
| | | 8 | PIC0 | All registers in PIC0 |
| | | 9 | TAPA0 | All registers in TAPA0 |
| | | 10 | ENCA0 | All registers in ENCA0 |
| | | 11 | TAUJ1 | All registers in TAUJ1 |
| | | 12 | TAUB0 | All registers in TAUB0 |
| | | 13 | TAUB1 | All registers in TAUB1 |
| | | 14 | PWM-Diag | All registers in PWM-Diag |

Table 33.36 PBG Groups and Channels and Target Modules (3/4)

| PBG Group | Group No. | PBG Channel Number | Protection Target Module | Target Register |
|---|---|---|---|---|
| PBG21 | 09 | 0 | Flash memory (data flash) | EEPRDCYCL |
| | | 1 | Flash memory (data flash ECC) | DFECCCTL, DFERSTR, DFERSTC, DFOVFSTR, DFOVFSTC, DFERRINT, DFEADR, DFTSTCTL |
| | | 2 | Reserved area | — |
| | | 3 | Reserved area | — |
| | | 4 | Reserved area | — |
| PBG30 | 03 | 0 | RSCAN0_CAN0 | All registers in RSCAN0 Ch0 group |
| | | 1 | RSCAN0_CAN1 | All registers in RSCAN0 Ch1 group |
| | | 2 | RSCAN0_CAN2 | All registers in RSCAN0 Ch2 group |
| | | 3 | RSCAN0_CAN3 | All registers in RSCAN0 Ch3 group |
| | | 4 | RSCAN0_CAN4 | All registers in RSCAN0 Ch4 group |
| | | 5 | RSCAN0_CAN5 | All registers in RSCAN0 Ch5 group |
| | | 6 | RSCAN0_Global | All registers in RSCAN0 Global group |
| | | 7 | RSCAN1_CAN6 | All registers in RSCAN1 Ch6 group |
| | | 8 | Reserved area | — |
| | | 9 | RSCAN1_Global | All registers in RSCAN1 Global group |
| | | 10 | Reserved area | — |
| | | 11 | Reserved area | — |
| | | 12 | ADCA1 | All registers in ADCA1 |
| PBG31 | 04 | 0 | OSTM0 | All registers in OSTM0 |
| | | 1 | OSTM1-4 | All registers in OSTM1, OSTM2, OSTM3, and OSTM4 |
| | | 2 | Reserved area | — |
| | | 3 | Reserved area | — |
| | | 4 | ECCCSIH0 | All registers in ECC CSIH0 |
| | | 5 | ECCCSIH1 | All registers in ECC CSIH1 |
| | | 6 | ECCCSIH2 | All registers in ECC CSIH2 |
| | | 7 | ECCCSIH3 | All registers in ECC CSIH3 |
| | | 8 | Reserved area | — |
| | | 9 | ECCCAN1 | All register in ECCCAN1 group |
| | | 10 | ECCCAN0 PHY1 | All registers in ECCCAN0 PHY1 group |
| | | 11 | ECCCAN0 PHY2 | All registers in ECCCAN0 PHY2 group |

Table 33.36 PBG Groups and Channels and Target Modules (4/4)

| PBG Group | Group No. | PBG Channel Number | Protection Target Module | Target Register |
|---|---|---|---|---|
| PBG32 | 05 | 0 | CSIH0 (Group A) | CSIH0CTL0-2, CSIH0STR0, CSIH0STCR0, CSIH0EMU |
| | | 1 | CSIH0 (Group B) | CSIH0 registers other than the above |
| | | 2 | CSIH1 (Group A) | CSIH1CTL0-2, CSIH1STR0, CSIH1STCR0, CSIH1EMU |
| | | 3 | CSIH1 (Group B) | CSIH1 registers other than the above |
| | | 4 | CSIH2 (Group A) | CSIH2CTL0-2, CSIH2STR0, CSIH2STCR0, CSIH2EMU |
| | | 5 | CSIH2 (Group B) | CSIH2 registers other than the above |
| | | 6 | CSIH3 (Group A) | CSIH3CTL0-2, CSIH3STR0, CSIH3STCR0, CSIH3EMU |
| | | 7 | CSIH3 (Group B) | CSIH3 registers other than the above |
| | | 8 | CSIG0 (Group A) | CSIG0CTL0-2, CSIG0STR0, CSIG0STCR0, CSIG0EMU |
| | | 9 | CSIG0 (Group B) | CSIG0 registers other than the above |
| | | 10 | CSIG1 (Group A) | CSIG1CTL0-2, CSIG1STR0, CSIG1STCR0, CSIG1EMU |
| | | 11 | CSIG1 (Group B) | CSIG1 registers other than the above |
| PBG50 | 06 | 0 | System control | All registers in Write-Protect Function, Reset Controller, Power Supply Circuit, Supply Voltage Monitor, Clock Controller, Clock Monitor, Stand-By Controller, and Low-Power Sampler*1 (except STBC0PSC, STBC0STPT, SWRESA, PROTCMD0, PROTS0) |
| | | 1 | STBC | STBC0PSC, STBC0STPT |
| | | 2 | Reserved area | — |
| | | 3 | Reserved area | — |
| | | 4 | Software reset | SWRESA |
| | | 5 | Flash memory (Self Programming) | —*2 |
| | | 6 | Flash memory (Control) | —*2 |
| | | 7 | Flash memory (Option Bytes) | OPBT0<br>PRDNAME1-3 |
| | | 8 | Write-Protect command registers | PROTCMD0<br>PROTS0 |

Note 1. For details, see **Section 5, Write-Protected Registers**, **Section 9, Reset Controller**, **Section 10, Power Supply Circuit**, **Section 11, Supply Voltage Monitor**, **Section 12, Clock Controller**, **Section 13, Clock Monitor (CLMA)**, **Section 14, Stand-By Controller (STBC)**, and **Section 15, Low-Power Sampler (LPS)**.

Note 2. Regarding the PBG registers for the flash memory, see the *RH850/F1K Flash Memory User's Manual:Hardware Interface*.

Note 3. Regarding the PBG register addresses, see the **Table 33.37, List of PBG Protection Registers**.

Note 4. Regarding the RSCAN and ECCCAN guard group, see the **Section 21, CAN Interface (RS-CAN)**.

### 33.3.2.1 List of Registers

The following table lists the registers provided for each PBG group. And PBG group is equal to module name.

**Table 33.37 List of PBG Protection Registers (1/4)**

| PBG Group | Group No. | Register Symbol | Register Name | R/W | Value after Reset | Address | Access Size | Power Domain |
|---|---|---|---|---|---|---|---|---|
| PBG10 | 00 | FSGD00PROT0 | PBG00 protection register 0 | R/W | 066F FFF7$_H$ | FFC4 0000$_H$ | 8/16/32 | ISO |
| | | FSGD00PROT1 | PBG00 protection register 1 | R/W | 066F FFF7$_H$ | FFC4 0004$_H$ | 8/16/32 | |
| | | FSGD00PROT2 | PBG00 protection register 2 | R/W | 066F FFF7$_H$ | FFC4 0008$_H$ | 8/16/32 | |
| | | FSGD00PROT3 | PBG00 protection register 3 | R/W | 066F FFF7$_H$ | FFC4 000C$_H$ | 8/16/32 | |
| | | FSGD00PROT4 | PBG00 protection register 4 | R/W | 066F FFF7$_H$ | FFC4 0010$_H$ | 8/16/32 | |
| | | FSGD00PROT5 | PBG00 protection register 5 | R/W | 066F FFF7$_H$ | FFC4 0014$_H$ | 8/16/32 | |
| | | FSGD00PROT6 | PBG00 protection register 6 | R/W | 066F FFF7$_H$ | FFC4 0018$_H$ | 8/16/32 | |
| | | FSGD00PROT7 | PBG00 protection register 7 | R/W | 066F FFF7$_H$ | FFC4 001C$_H$ | 8/16/32 | |
| | | FSGD00PROT8 | PBG00 protection register 8 | R/W | 066F FFF7$_H$ | FFC4 0020$_H$ | 8/16/32 | |
| | | FSGD00PROT9 | PBG00 protection register 9 | R/W | 066F FFF7$_H$ | FFC4 0024$_H$ | 8/16/32 | |
| | | FSGD00PROT10 | PBG00 protection register 10 | R/W | 066F FFF7$_H$ | FFC4 0028$_H$ | 8/16/32 | |
| | | FSGD00PROT11 | PBG00 protection register 11 | R/W | 066F FFF7$_H$ | FFC4 002C$_H$ | 8/16/32 | |
| | | FSGD00PROT12 | PBG00 protection register 12 | R/W | 066F FFF7$_H$ | FFC4 0030$_H$ | 8/16/32 | |
| | | FSGD00PROT13 | PBG00 protection register 13 | R/W | 066F FFF7$_H$ | FFC4 0034$_H$ | 8/16/32 | |
| PBG11 | 01 | FSGD01PROT0 | PBG01 protection register 0 | R/W | 066F FFF7$_H$ | FFC4 0100$_H$ | 8/16/32 | ISO |
| | | FSGD01PROT1 | PBG01 protection register 1 | R/W | 066F FFF7$_H$ | FFC4 0104$_H$ | 8/16/32 | |
| | | FSGD01PROT2 | PBG01 protection register 2 | R/W | 066F FFF7$_H$ | FFC4 0108$_H$ | 8/16/32 | |
| | | FSGD01PROT3 | PBG01 protection register 3 | R/W | 066F FFF7$_H$ | FFC4 010C$_H$ | 8/16/32 | |
| | | FSGD01PROT4 | PBG01 protection register 4 | R/W | 066F FFF7$_H$ | FFC4 0110$_H$ | 8/16/32 | |
| | | FSGD01PROT5 | PBG01 protection register 5 | R/W | 066F FFF7$_H$ | FFC4 0114$_H$ | 8/16/32 | |
| | | FSGD01PROT6 | PBG01 protection register 6 | R/W | 066F FFF7$_H$ | FFC4 0118$_H$ | 8/16/32 | |
| | | FSGD01PROT7 | PBG01 protection register 7 | R/W | 066F FFF7$_H$ | FFC4 011C$_H$ | 8/16/32 | |
| | | FSGD01PROT8 | PBG01 protection register 8 | R/W | 066F FFF7$_H$ | FFC4 0120$_H$ | 8/16/32 | |
| | | FSGD01PROT9 | PBG01 protection register 9 | R/W | 066F FFF7$_H$ | FFC4 0124$_H$ | 8/16/32 | |
| | | FSGD01PROT10 | PBG01 protection register 10 | R/W | 066F FFF7$_H$ | FFC4 0128$_H$ | 8/16/32 | |
| | | — | Reserved | — | — | FFC4 012C$_H$ | — | |
| | | FSGD01PROT12 | PBG01 protection register 12 | R/W | 066F FFF7$_H$ | FFC4 0130$_H$ | 8/16/32 | |
| | | FSGD01PROT13 | PBG01 protection register 13 | R/W | 066F FFF7$_H$ | FFC4 0134$_H$ | 8/16/32 | |
| | | FSGD01PROT14 | PBG01 protection register 14 | R/W | 066F FFF7$_H$ | FFC4 0138$_H$ | 8/16/32 | |

Table 33.37 List of PBG Protection Registers (2/4)

| PBG Group | Group No. | Register Symbol | Register Name | R/W | Value after Reset | Address | Access Size | Power Domain |
|---|---|---|---|---|---|---|---|---|
| PBG12 | 07 | FSGD07PROT0 | PBG07 protection register 0 | R/W | 066F FFF7$_H$ | FFC4 0400$_H$ | 8/16/32 | ISO |
| | | FSGD07PROT1 | PBG07 protection register 1 | R/W | 066F FFF7$_H$ | FFC4 0404$_H$ | 8/16/32 | |
| | | FSGD07PROT2 | PBG07 protection register 2 | R/W | 066F FFF7$_H$ | FFC4 0408$_H$ | 8/16/32 | |
| | | FSGD07PROT3 | PBG07 protection register 3 | R/W | 066F FFF7$_H$ | FFC4 040C$_H$ | 8/16/32 | |
| | | FSGD07PROT4 | PBG07 protection register 4 | R/W | 066F FFF7$_H$ | FFC4 0410$_H$ | 8/16/32 | |
| | | FSGD07PROT5 | PBG07 protection register 5 | R/W | 066F FFF7$_H$ | FFC4 0414$_H$ | 8/16/32 | |
| | | FSGD07PROT6 | PBG07 protection register 6 | R/W | 066F FFF7$_H$ | FFC4 0418$_H$ | 8/16/32 | |
| | | FSGD07PROT7 | PBG07 protection register 7 | R/W | 066F FFF7$_H$ | FFC4 041C$_H$ | 8/16/32 | |
| | | FSGD07PROT8 | PBG07 protection register 8 | R/W | 066F FFF7$_H$ | FFC4 0420$_H$ | 8/16/32 | |
| | | FSGD07PROT9 | PBG07 protection register 9 | R/W | 066F FFF7$_H$ | FFC4 0424$_H$ | 8/16/32 | |
| | | FSGD07PROT10 | PBG07 protection register 10 | R/W | 066F FFF7$_H$ | FFC4 0428$_H$ | 8/16/32 | |
| | | FSGD07PROT11 | PBG07 protection register 11 | R/W | 066F FFF7$_H$ | FFC4 042C$_H$ | 8/16/32 | |
| PBG12 | 07 | FSGD07PROT12 | PBG07 protection register 12 | R/W | 066F FFF7$_H$ | FFC4 0430$_H$ | 8/16/32 | ISO |
| PBG13 | 08 | FSGD08PROT0 | PBG08 protection register 0 | R/W | 066F FFF7$_H$ | FFC4 0500$_H$ | 8/16/32 | ISO |
| | | FSGD08PROT1 | PBG08 protection register 1 | R/W | 066F FFF7$_H$ | FFC4 0504$_H$ | 8/16/32 | |
| | | FSGD08PROT2 | PBG08 protection register 2 | R/W | 066F FFF7$_H$ | FFC4 0508$_H$ | 8/16/32 | |
| | | FSGD08PROT3 | PBG08 protection register 3 | R/W | 066F FFF7$_H$ | FFC4 050C$_H$ | 8/16/32 | |
| | | FSGD08PROT4 | PBG08 protection register 4 | R/W | 066F FFF7$_H$ | FFC4 0510$_H$ | 8/16/32 | |
| | | FSGD08PROT5 | PBG08 protection register 5 | R/W | 066F FFF7$_H$ | FFC4 0514$_H$ | 8/16/32 | |
| PBG20 | 02 | FSGD02PROT0 | PBG02 protection register 0 | R/W | 066F FFF7$_H$ | FFDD D000$_H$ | 8/16/32 | ISO |
| | | FSGD02PROT1 | PBG02 protection register 1 | R/W | 066F FFF7$_H$ | FFDD D004$_H$ | 8/16/32 | |
| | | FSGD02PROT2 | PBG02 protection register 2 | R/W | 066F FFF7$_H$ | FFDD D008$_H$ | 8/16/32 | |
| | | FSGD02PROT3 | PBG02 protection register 3 | R/W | 066F FFF7$_H$ | FFDD D00C$_H$ | 8/16/32 | |
| | | FSGD02PROT4 | PBG02 protection register 4 | R/W | 066F FFF7$_H$ | FFDD D010$_H$ | 8/16/32 | |
| | | FSGD02PROT5 | PBG02 protection register 5 | R/W | 066F FFF7$_H$ | FFDD D014$_H$ | 8/16/32 | |
| | | FSGD02PROT6 | PBG02 protection register 6 | R/W | 0607 FE77$_H$ | FFDD D018$_H$ | 8/16/32 | |
| | | — | Reserved | — | — | FFDD D01C$_H$ | — | |
| | | FSGD02PROT8 | PBG02 protection register 8 | R/W | 066F FFF7$_H$ | FFDD D020$_H$ | 8/16/32 | |
| | | FSGD02PROT9 | PBG02 protection register 9 | R/W | 066F FFF7$_H$ | FFDD D024$_H$ | 8/16/32 | |
| | | FSGD02PROT10 | PBG02 protection register 10 | R/W | 066F FFF7$_H$ | FFDD D028$_H$ | 8/16/32 | |
| | | FSGD02PROT11 | PBG02 protection register 11 | R/W | 066F FFF7$_H$ | FFDD D02C$_H$ | 8/16/32 | |
| | | FSGD02PROT12 | PBG02 protection register 12 | R/W | 066F FFF7$_H$ | FFDD D030$_H$ | 8/16/32 | |
| | | FSGD02PROT13 | PBG02 protection register 13 | R/W | 066F FFF7$_H$ | FFDD D034$_H$ | 8/16/32 | |
| | | FSGD02PROT14 | PBG02 protection register 14 | R/W | 066F FFF7$_H$ | FFDD D038$_H$ | 8/16/32 | |
| PBG21 | 09 | FSGD09PROT0 | PBG09 protection register 0 | R/W | 066F FFF7$_H$ | FFDD D100$_H$ | 8/16/32 | ISO |
| | | FSGD09PROT1 | PBG09 protection register 1 | R/W | 066F FFF7$_H$ | FFDD D104$_H$ | 8/16/32 | |
| | | — | Reserved | — | — | FFDD D108$_H$ | — | |
| | | — | Reserved | — | — | FFDD D10C$_H$ | — | |
| | | — | Reserved | — | — | FFDD D110$_H$ | — | |

Table 33.37 List of PBG Protection Registers (3/4)

| PBG Group | Group No. | Register Symbol | Register Name | R/W | Value after Reset | Address | Access Size | Power Domain |
|---|---|---|---|---|---|---|---|---|
| PBG30 | 03 | FSGD03PROT0 | PBG03 protection register 0 | R/W | 066F FFF7$_H$ | FFF9 4000$_H$ | 8/16/32 | ISO |
| | | FSGD03PROT1 | PBG03 protection register 1 | R/W | 066F FFF7$_H$ | FFF9 4004$_H$ | 8/16/32 | |
| | | FSGD03PROT2 | PBG03 protection register 2 | R/W | 066F FFF7$_H$ | FFF9 4008$_H$ | 8/16/32 | |
| | | FSGD03PROT3 | PBG03 protection register 3 | R/W | 066F FFF7$_H$ | FFF9 400C$_H$ | 8/16/32 | |
| | | FSGD03PROT4 | PBG03 protection register 4 | R/W | 066F FFF7$_H$ | FFF9 4010$_H$ | 8/16/32 | |
| | | FSGD03PROT5 | PBG03 protection register 5 | R/W | 066F FFF7$_H$ | FFF9 4014$_H$ | 8/16/32 | |
| | | FSGD03PROT6 | PBG03 protection register 6 | R/W | 066F FFF7$_H$ | FFF9 4018$_H$ | 8/16/32 | |
| | | FSGD03PROT7 | PBG03 protection register 7 | R/W | 066F FFF7$_H$ | FFF9 401C$_H$ | 8/16/32 | |
| | | — | Reserved | — | — | FFF9 4020$_H$ | — | |
| | | FSGD03PROT9 | PBG03 protection register 9 | R/W | 066F FFF7$_H$ | FFF9 4024$_H$ | 8/16/32 | |
| | | — | Reserved | — | — | FFF9 4028$_H$ | — | |
| | | — | Reserved | — | — | FFF9 402C$_H$ | — | |
| | | FSGD03PROT12 | PBG03 protection register 12 | R/W | 066F FFF7$_H$ | FFF9 4030$_H$ | 8/16/32 | |
| PBG31 | 04 | FSGD04PROT0 | PBG04 protection register 0 | R/W | 066F FFF7$_H$ | FFF9 4100$_H$ | 8/16/32 | ISO |
| | | FSGD04PROT1 | PBG04 protection register 1 | R/W | 0607 FE77$_H$ | FFF9 4104$_H$ | 8/16/32 | |
| | | — | Reserved | — | — | FFF9 4108$_H$ | — | |
| PBG31 | 04 | — | Reserved | — | — | FFF9 410C$_H$ | — | ISO |
| | | FSGD04PROT4 | PBG04 protection register 4 | R/W | 066F FFF7$_H$ | FFF9 4110$_H$ | 8/16/32 | |
| | | FSGD04PROT5 | PBG04 protection register 5 | R/W | 066F FFF7$_H$ | FFF9 4114$_H$ | 8/16/32 | |
| | | FSGD04PROT6 | PBG04 protection register 6 | R/W | 066F FFF7$_H$ | FFF9 4118$_H$ | 8/16/32 | |
| | | FSGD04PROT7 | PBG04 protection register 7 | R/W | 066F FFF7$_H$ | FFF9 411C$_H$ | 8/16/32 | |
| | | — | Reserved | — | — | FFF9 4120$_H$ | — | |
| | | FSGD04PROT9 | PBG04 protection register 9 | R/W | 066F FFF7$_H$ | FFF9 4124$_H$ | 8/16/32 | |
| | | FSGD04PROT10 | PBG04 protection register 10 | R/W | 066F FFF7$_H$ | FFF9 4128$_H$ | 8/16/32 | |
| | | FSGD04PROT11 | PBG04 protection register 11 | R/W | 066F FFF7$_H$ | FFF9 412C$_H$ | 8/16/32 | |
| PBG32 | 05 | FSGD05PROT0 | PBG05 protection register 0 | R/W | 066F FFF7$_H$ | FFF9 4200$_H$ | 8/16/32 | ISO |
| | | FSGD05PROT1 | PBG05 protection register 1 | R/W | 066F FFF7$_H$ | FFF9 4204$_H$ | 8/16/32 | |
| | | FSGD05PROT2 | PBG05 protection register 2 | R/W | 066F FFF7$_H$ | FFF9 4208$_H$ | 8/16/32 | |
| | | FSGD05PROT3 | PBG05 protection register 3 | R/W | 066F FFF7$_H$ | FFF9 420C$_H$ | 8/16/32 | |
| | | FSGD05PROT4 | PBG05 protection register 4 | R/W | 066F FFF7$_H$ | FFF9 4210$_H$ | 8/16/32 | |
| | | FSGD05PROT5 | PBG05 protection register 5 | R/W | 066F FFF7$_H$ | FFF9 4214$_H$ | 8/16/32 | |
| | | FSGD05PROT6 | PBG05 protection register 6 | R/W | 066F FFF7$_H$ | FFF9 4218$_H$ | 8/16/32 | |
| | | FSGD05PROT7 | PBG05 protection register 7 | R/W | 066F FFF7$_H$ | FFF9 421C$_H$ | 8/16/32 | |
| | | FSGD05PROT8 | PBG05 protection register 8 | R/W | 066F FFF7$_H$ | FFF9 4220$_H$ | 8/16/32 | |
| | | FSGD05PROT9 | PBG05 protection register 9 | R/W | 066F FFF7$_H$ | FFF9 4224$_H$ | 8/16/32 | |
| | | FSGD05PROT10 | PBG05 protection register 10 | R/W | 066F FFF7$_H$ | FFF9 4228$_H$ | 8/16/32 | |
| | | FSGD05PROT11 | PBG05 protection register 11 | R/W | 066F FFF7$_H$ | FFF9 422C$_H$ | 8/16/32 | |

Table 33.37 List of PBG Protection Registers (4/4)

| PBG Group | Group No. | Register Symbol | Register Name | R/W | Value after Reset | Address | Access Size | Power Domain |
|---|---|---|---|---|---|---|---|---|
| PBG50 | 06 | FSGD06PROT0 | PBG06 protection register 0 | R/W | 066F FFF7$_H$ | FFF9 0000$_H$ | 8/16/32 | ISO |
| | | FSGD06PROT1 | PBG06 protection register 1 | R/W | 0647 FF77$_H$ | FFF9 0004$_H$ | 8/16/32 | |
| | | — | Reserved | — | — | FFF9 0008$_H$ | — | |
| | | — | Reserved | — | — | FFF9 000C$_H$ | — | |
| | | FSGD06PROT4 | PBG06 protection register 4 | R/W | 066F FFF7$_H$ | FFF9 0010$_H$ | 8/16/32 | |
| | | FSGD06PROT5 | PBG06 protection register 5 | R/W | 066F FFF7$_H$ | FFF9 0014$_H$ | 8/16/32 | |
| | | — | Reserved | — | — | FFF9 0018$_H$ | — | |
| | | FSGD06PROT7 | PBG06 protection register 7 | R/W | 066F FFF7$_H$ | FFF9 001C$_H$ | 8/16/32 | |
| | | FSGD06PROT8 | PBG06 protection register 8 | R/W | 066F FFF7$_H$ | FFF9 0020$_H$ | 8/16/32 | |

The following table lists the registers provided for each PBG group.

**Table 33.38 List of PBG Error Registers**

| Module Name | Register Symbol | Register Name | R/W | Value after Reset | Address | Access Size |
|---|---|---|---|---|---|---|
| PBGxx | ERRSLVxxCTL | PBGxx error control register | W | 0000 $0000_H$ | <base_addr0> + $0_H$ | 8/16/32 |
| | ERRSLVxxSTAT | PBGxx error status register | R | 0000 $0000_H$ | <base_addr0> + $4_H$ | 8/16/32 |
| | ERRSLVxxADDR | PBGxx error address register | R | 0000 $0000_H$ | <base_addr0> + $8_H$ | 32 |
| | ERRSLVxxTYPE | PBGxx error type register | R | 0000 $0000_H$ | <base_addr0> + $C_H$ | 16/32 |

In the above table, “xx” in the register names and symbols represents the PBG group numbers. The table below shows the base address values <base_addr0>, which correspond to each of the PBG group numbers.

**Table 33.39 PBG Group Numbers and Error Base Addresses**

| PBG Group | PBG Group Number | <base_addr0> |
|---|---|---|
| PBG10 | 00 | FFC4 $0040_H$ |
| PBG11 | 01 | FFC4 $0140_H$ |
| PBG12 | 07 | FFC4 $0440_H$ |
| PBG13 | 08 | FFC4 $0540_H$ |
| PBG20 | 02 | FFDD $D040_H$ |
| PBG21 | 09 | FFDD $D140_H$ |
| PBG30 | 03 | FFF9 $4040_H$ |
| PBG31 | 04 | FFF9 $4140_H$ |
| PBG32 | 05 | FFF9 $4240_H$ |
| PBG50 | 06 | FFF9 $0040_H$ |

### 33.3.2.2 Details of Registers

#### (1) FSGDxxPROTn — PBGxx Protection Register n

FSGDxxPROTn specifies the access to be rejected for protecting the target peripheral circuit control registers. Any access that is disabled with any of the identifiers is rejected as an illegal access. "n" in the register names and symbols represents the PBG channel number.

**Access:** FSGDxxPROTn register can be read or written in 32-bit units.
FSGDxxPROTnL and FSGDxxPROTnH registers can be read or written in 16-bit units.
FSGDxxPROTnLL, FSGDxxPROTnLH, FSGDxxPROTnHL, and FSGDxxPROTnHH registers can be read or written in 8-bit units.

**Address:** See **Table 33.37, List of PBG Protection Registers**

**Value after reset:** See **Table 33.37, List of PBG Protection Registers**

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROTLOCK | — | — | — | — | — | PROTUM | PROTPEID[7:0] | | | | | | | | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | 1 |
| R/W | R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | | | | | | | PROTSPID[3:0] | | | | — | PROTRDPDEF | PROTWRPDEF | PROTRD | PROTWR |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | *1 | *1 | *1 | *1 | 1 | 0 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W |

Note 1. It varies depending on each register. See **Section 33.3.2.1, List of Registers**.

**Table 33.40 FSGDxxPROTn Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | PROTLOCK | Register Lock<br>0: Enables FSGDxxPROTn rewrite.<br>1: Disables FSGDxxPROTn rewrite.<br>When PROTLOCK is set to 1, the value is held until reset is asserted. |
| 30 to 26 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 25 | PROTUM | User Mode Access<br>0: Enables access in supervisor mode.<br>1: Enables access in user mode and supervisor mode. |
| 24 to 17 | PROTPEID[7:0] | PEID Access*1<br>The PROTPEID field is a bit list, in which one bit corresponds to one PEID value. Setting multiple bits enables multiple ID values simultaneously. For example, setting the PROTPEID field to 0000 $0110_B$ enables access with PEID = 1 and PEID = 2.<br>0: Disables access with PEIDn.<br>1: Enables access with PEIDn. |
| 16 to 9 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 8 to 5 | PROTSPID[3:0] | SPID Access*2<br>The PROTSPID field is a bit list, in which one bit corresponds to one SPID value. Setting multiple bits enables multiple SPID values simultaneously. For example, setting the PROTSPID field to $0110_B$ enables access with SPID = 1 and SPID = 2.<br>0: Disables access with SPIDn.<br>1: Enables access with SPIDn. |
| 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |

**Table 33.40 FSGDxxPROTn Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 3 | PROTRDPDEF | Default Read Protection<br>0: Enables read access from any master.<br>1: Only enables read access from the master having passed the filter. |
| 2 | PROTWRPDEF | Default Write Protection<br>0: Enables write access from any master.<br>1: Only enables write access from the master having passed the filter. |
| 1 | PROTRD | Read Permission<br>0: Disables reading by a bus master subject to access filtering.<br>1: Enables reading by a bus master subject to access filtering. |
| 0 | PROTWR | Write Permission<br>0: Disables writing by a bus master subject to access filtering.<br>1: Enables writing by a bus master subject to access filtering. |

Note 1. Access with PEID
PROTPEID is a bit list with each bit corresponding to a PEID value.
Setting multiple bits enables ID values of multiple bus masters at the same time.

Note 2. Access with SPID
PROTSPID is a bit list with each bit representing an SPID value.
Setting multiple bits enables ID values of multiple masters at the same time.

### (2) ERRSLVxxCTL — PBGxx Error Control Register

ERRSLVxxCTL clears the status in error status register PBGxx.

**Access:** ERRSLVxxCTL register is a write only register that can be written in 32-bit units.
ERRSLVxxCTLL register is a write only register that can be written in 16-bit units.
ERRSLVxxCTLLL register is a write only register that can be written in 8-bit units.

**Address:** ERRSLVxxCTL: <base_addr0> + $0_H$
ERRSLVxxCTLL: <base_addr0> + $0_H$
ERRSLVxxCTLLL: <base_addr0> + $0_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CLRO | CLRE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W | W |

**Table 33.41 ERRSLVxxCTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 1 | CLRO | Clears the overflow flag.<br>0: Does not clear the overflow flag.<br>1: Clears the overflow flag. |
| 0 | CLRE | Clears the error flag.<br>0: Does not clear the error flag.<br>1: Clears the error flag. |

| CLRO | CLRE | Function |
|---|---|---|
| 0 | 0 | Clears neither of the bits. |
| 0 | 1 | Setting prohibited |
| 1 | 0 | Clears the OVF bit. |
| 1 | 1 | Clears the OVF and ERR bits. |

### (3) ERRSLVxxSTAT — PBGxx Error Status Register

ERRSLVxxSTAT holds the status of the illegal access rejected with the PBGxx.

**Access:** ERRSLVxxSTAT register is a read only register that can be read in 32-bit units.
ERRSLVxxSTATL register is a read only register that can be read in 16-bit units.
ERRSLVxxSTATLL register is a read only register that can be read in 8-bit units.

**Address:** ERRSLVxxSTAT: <base_addr0> + $4_H$
ERRSLVxxSTATL: <base_addr0> + $4_H$
ERRSLVxxSTATLL: <base_addr0> + $4_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OVF | ERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 33.42 ERRSLVxxSTAT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1 | OVF | Error Entry Overflow Flag<br>0: No overflow<br>1: An overflow occurred.<br>If a second guard violation occurs with the error detection flag being set after the first guard violation occurs, the error entry overflows and this flag is set because the number of PBG error entry stages is 1.<br>Note that this overflow is not notified to INTGUARD. The error information of the guard violation when an overflow occurs are not captured. |
| 0 | ERR | Error Status Flag<br>0: No PBG protection violation<br>1: A PBG protection violation occurred. |

### (4) ERRSLVxxADDR — PBGxx Error Address Register

ERRSLVxxADDR holds the address of the illegal access rejected with the PBGxx. The register is not updated when corresponding ERRSLVxxSTAT.ERR is 1.

**Access:** ERRSLVxxADDR register is a read only register that can be read in 32-bit units.

**Address:** <base_addr0> + $8_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADDR[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADDR[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 33.43 ERRSLVxxADDR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | ADDR[31:2] | Address in which the PBG protection violation is generated. |
| 1, 0 | Reserved | When read, the value after reset is returned. |

### (5) ERRSLVxxTYPE — PBGxx Error Type Register

ERRSLVxxTYPE holds the type of the illegal access rejected with the PBGxx.

**Access:** ERRSLVxxTYPE register is a read only register that can be read in 32-bit units.
ERRSLVxxTYPEL register is a read only register that can be read in 16-bit units.

**Address:** ERRSLVxxTYPE: <base_addr0> + $C_H$
ERRSLVxxTYPEL: <base_addr0> + $C_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PEID[2:0] | | | — | — | — | SPID[1:0] | | — | UM | — | — | — | — | — | WRITE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 33.44 ERRSLVxxTYPE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. |
| 15 to 13 | PEID[2:0] | PEID of the access source from which the PBG protection violation is generated. |
| 12 to 10 | Reserved | When read, the value after reset is returned. |
| 9, 8 | SPID[1:0] | SPID of the access source from which the PBG protection violation is generated. |
| 7 | Reserved | When read, the value after reset is returned. |
| 6 | UM | UM of the access source from which the PBG protection violation is generated. |
| 5 | Reserved | When read, the value after reset is returned. |
| 4 to 1 | Reserved | These bits are read as an undefined value. |
| 0 | WRITE | This bit is set to 1 when the access that has generated the PBG protection violation is a write access. |

### 33.3.3 PBG for CPU System

The PBGC module is divided into two PBGC groups, PBGC0 and PBGC1. PBGC0 group contains protection registers for INTC2 and DMA functions. PBGC1 group contains protection registers for ECC control function etc. Each PBGC group holds the information of the access that has been rejected.

The following table lists the target registers to be protected and the corresponding PBGC group names.

**Table 33.45 Target Registers of PBG for CPU Subsystem**

| PBGC Group | Group No. | Channel Number | Protection Target Module | Target Register |
|---|---|---|---|---|
| PBGC0 | C0<br>(PBC group 0 for CPU system) | 0 | INTC2 | ICxxx (xxx=32 to 357)<br>IMRm (m=1 to 11)<br>(described in **Section 7, Exception/ Interrupts**) |
| | | 1 | DMA | All registers inside DMA controller<br>(described in **Section 8, DMA Controller**) |
| PBGC1 | C1<br>(PBC group 1 for CPU system) | 0 | Flash memory<br>(Programing function) | BFASELR*1 |
| | | 1 | Code flash ECC control register (VCI) | CFECCCTL_VCI<br>CFERRINT_VCI<br>CFSTCLR_VCI<br>CFOVFSTR_VCI<br>CF1STERSTR_VCI<br>CF1STEADR0_VCI<br>CFSTSTCTL_VCI |
| | | 2 | Code flash ECC control register (PE1) | CFECCCTL_PE1<br>CFERRINT_PE1<br>CFSTCLR_PE1<br>CFOVFSTR_PE1<br>CF1STERSTR_PE1<br>CF1STEADR0_PE1<br>CFSTSTCTL_PE1 |
| | | 3 | Local RAM ECC control register (PE1) | LRTSTCTL_PE1<br>LRTDATBF0_PE1<br>LRECCCTL_PE1<br>LRERRINT_PE1<br>LRSTCLR_PE1<br>LROVFSTR_PE1<br>LR1STERSTR_PE1<br>LR1STEADR0_PE1 |
| | | 4 | On-Chip Debug module | EPC<br>(described in **Section 36, On-Chip Debug Unit (OCD)**) |
| | | 5 | Buffer controller | FBUFCCTL<br>(described in **Section 3, CPU System**) |

Note 1. Regarding the PBGC registers for the flash memory, see the *RH850/F1K Flash Memory User's Manual:Hardware Interface*.

### 33.3.3.1 List of Registers

The following table lists the registers provided for each PBGC group. And PBG group is equal to module name.

**Table 33.46 List of PBGC Protection Registers**

| PBG Group | Group No. | Register Symbol | Register Name | R/W | Value after Reset | Address | Access Size | Power Domain |
|---|---|---|---|---|---|---|---|---|
| PBGC0 | C0 | FSGDC0PROT0 | PBGC0 protection register 0 | R/W | 07FF FFFF$_H$ | FFC4 C000$_H$ | 8/16/32 | ISO |
| | | FSGDC0PROT1 | PBGC0 protection register 1 | R/W | 07FF FFFF$_H$ | FFC4 C004$_H$ | 8/16/32 | |
| PBGC1 | C1 | FSGDC1PROT0 | PBGC1 protection register 0 | R/W | 07FF FFFF$_H$ | FFC4 C120$_H$ | 8/16/32 | ISO |
| | | FSGDC1PROT1 | PBGC1 protection register 1 | R/W | 07FF FFFF$_H$ | FFC4 C124$_H$ | 8/16/32 | |
| | | FSGDC1PROT2 | PBGC1 protection register 2 | R/W | 07FF FFFF$_H$ | FFC4 C128$_H$ | 8/16/32 | |
| | | FSGDC1PROT3 | PBGC1 protection register 3 | R/W | 07FF FFFF$_H$ | FFC4 C12C$_H$ | 8/16/32 | |
| | | FSGDC1PROT4 | PBGC1 protection register 4 | R/W | 07FF FFFF$_H$ | FFC4 C130$_H$ | 8/16/32 | |
| | | FSGDC1PROT5 | PBGC1 protection register 5 | R/W | 07FF FFFF$_H$ | FFC4 C134$_H$ | 8/16/32 | |

**Table 33.47 List of PBGC Error Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| PBGC0 | PBGC0 error control register | ERRSLVC0CTL | FFC4 C800$_H$ |
| | PBGC0 error status register | ERRSLVC0STAT | FFC4 C804$_H$ |
| | PBGC0 error address register | ERRSLVC0ADDR | FFC4 C808$_H$ |
| | PBGC0 error type register | ERRSLVC0TYPE | FFC4 C80C$_H$ |
| PBGC1 | PBGC1 error control register | ERRSLVC1CTL | FFC4 C900$_H$ |
| | PBGC1 error status register | ERRSLVC1STAT | FFC4 C904$_H$ |
| | PBGC1 error address register | ERRSLVC1ADDR | FFC4 C908$_H$ |
| | PBGC1 error type register | ERRSLVC1TYPE | FFC4 C90C$_H$ |

### 33.3.3.2 Details of Registers

#### (1) FSGDCxPROTn — PBGCx Protection Register n

FSGDCxPROTn specifies the access to be rejected for protecting the target registers. Any access that is disabled with any of the identifiers is rejected as an illegal access.

"n" in the register names and symbols represents the PBGC channel number.

**Access:** FSGDCxPROTn register can be read or written in 32-bit units.
FSGDCxPROTnL and FSGDCxPROTnH registers can be read or written in 16-bit units.
FSGDCxPROTnLL, FSGDCxPROTnLH, FSGDCxPROTnHL, and FSGDCxPROTnHH registers can be read or written in 8-bit units.

**Address:** See **Table 33.46, List of PBGC Protection Registers**

**Value after reset:** See **Table 33.46, List of PBGC Protection Registers**

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | PROTUM | PROTPEID[7:0] | | | | | | | | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | PROTSPID[3:0] | | | | — | PROTRDPDEF | PROTWRPDEF | PROTRD | PROTWR |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W |

**Table 33.48 FSGDCxPROTn Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 25 | PROTUM | User Mode Access<br>0: Enables access in supervisor mode.<br>1: Enables access in user mode and supervisor mode. |
| 24 to 17 | PROTPEID[7:0] | PEID Access*1<br>The PROTPEID field is a bit list, in which one bit corresponds to one PEID value. Setting multiple bits enables multiple ID values simultaneously. For example, setting the PROTPEID field to 0000 $0110_B$ enables access with PEID = 1 and PEID = 2.<br>0: Disables access with PEIDn.<br>1: Enables access with PEIDn. |
| 16 to 9 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 8 to 5 | PROTSPID[3:0] | SPID Access*2<br>The PROTSPID field is a bit list, in which one bit corresponds to one SPID value. Setting multiple bits enables multiple SPID values simultaneously. For example, setting the PROTSPID field to $0110_B$ enables access with SPID = 1 and SPID = 2.<br>0: Disables access with SPIDn.<br>1: Enables access with SPIDn. |
| 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 | PROTRDPDEF | Default Read Protection<br>0: Enables read access from any master.<br>1: Only enables read access from the master having passed the filter. |

**Table 33.48 FSGDCxPROTn Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 2 | PROTWRPDEF | Default Write Protection<br>0: Enables write access from any master.<br>1: Only enables write access from the master having passed the filter. |
| 1 | PROTRD | Read Permission<br>0: Disables reading by a bus master subject to access filtering.<br>1: Enables reading by a bus master subject to access filtering. |
| 0 | PROTWR | Write Permission<br>0: Disables writing by a bus master subject to access filtering.<br>1: Enables writing by a bus master subject to access filtering. |

Note 1. Access with PEID
PROTPEID is a bit list with each bit corresponding to a PEID value.
Setting multiple bits enables ID values of multiple bus masters at the same time.

Note 2. Access with SPID
PROTSPID is a bit list with each bit representing an SPID value.
Setting multiple bits enables ID values of multiple masters at the same time.

### (2) ERRSLVCxCTL — PBGCx Error Control Register

ERRSLVCxCTL clears the status in error status register PBGCx.

**Access:** ERRSLVCxCTL register is a write only register that can be written in 32-bit units.
ERRSLVCxCTLL register is a write only register that can be written in 16-bit units.
ERRSLVCxCTLLL register is a write only register that can be written in 8-bit units.

**Address:** ERRSLVCxCTL: FFC4 C800H + ($100_H$ × x)
ERRSLVCxCTLL: FFC4 C800H + ($100_H$ × x)
ERRSLVCxCTLLL: FFC4 C800H + ($100_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CLRO | CLRE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W | W |

**Table 33.49 ERRSLVCxCTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When writing, write the value after reset. |
| 1 | CLRO | Clears the overflow flag.<br>0: Does not clear the overflow flag.<br>1: Clears the overflow flag. |
| 0 | CLRE | Clears the error flag.<br>0: Does not clear the error flag.<br>1: Clears the error flag. |

| CLRO | CLRE | Function |
|---|---|---|
| 0 | 0 | Clears neither of the bits. |
| 0 | 1 | Setting prohibited |
| 1 | 0 | Clears the OVF bit. |
| 1 | 1 | Clears the OVF and ERR bits. |

### (3) ERRSLVCxSTAT — PBGCx Error Status Register

ERRSLVCxSTAT holds the status of the illegal access rejected with the PBGCx.

**Access:** ERRSLVCxSTAT register is a read only register that can be read in 32-bit units.
ERRSLVCxSTATL register is a read only register that can be read in 16-bit units.
ERRSLVCxSTATLL register is a read only register that can be read in 8-bit units.

**Address:** ERRSLVCxSTAT: FFC4 C804H + ($100_H$ × x)
ERRSLVCxSTATL: FFC4 C804H + ($100_H$ × x)
ERRSLVCxSTATLL: FFC4 C804H + ($100_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OVF | ERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 33.50 ERRSLVCxSTAT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is returned. |
| 1 | OVF | Error Entry Overflow Flag<br>0: No overflow<br>1: An overflow occurred.<br>If a second guard violation occurs with the error detection flag being set after the first guard violation occurs, the error entry overflows and this flag is set because the number of PBGC error entry stages is 1.<br>Note that this overflow is not notified to INTGUARD. The error information of the guard violation when an overflow occurs are not captured. |
| 0 | ERR | Error Status Flag<br>0: No PBG protection violation<br>1: A PBG protection violation occurred. |

### (4) ERRSLVCxADDR — PBGCx Error Address Register

ERRSLVCxADDR holds the address of the illegal access rejected with the PBGCx.

The register is not updated when corresponding ERRSLVCxSTAT.ERR is 1.

**Access:** ERRSLVCxADDR register is a read only register that can be read in 32-bit units.

**Address:** FFC4 C808H + ($100_H$ × x)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADDR[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADDR[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 33.51 ERRSLVCxADDR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | ADDR[31:2] | Address in which the PBG protection violation is generated. |
| 1, 0 | Reserved | When read, the value after reset is returned. |

### (5) ERRSLVCxTYPE — PBGCx Error Type Register

ERRSLVCxTYPE holds the type of the illegal access rejected with the PBGCx.

The register is not updated when corresponding ERRSLVCxSTAT.ERR is 1.

**Access:** ERRSLVCxTYPE register is a read only register that can be read in 32-bit units.
ERRSLVCxTYPEL register is a read only register that can be read in 16-bit units.

**Address:** ERRSLVCxTYPE: FFC4 C80CH + ($100_H$ × x)
ERRSLVCxTYPEL: FFC4 C80CH + ($100_H$ × x)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PEID[2:0] | | | — | — | — | SPID[1:0] | | — | UM | — | — | — | — | — | WRITE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 33.52 ERRSLVCxTYPE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is returned. |
| 15 to 13 | PEID[2:0] | PEID of the access source from which the PBG protection violation is generated. |
| 12 to 10 | Reserved | When read, the value after reset is returned. |
| 9, 8 | SPID[1:0] | SPID of the access source from which the PBG protection violation is generated. |
| 7 | Reserved | When read, the value after reset is returned. |
| 6 | UM | UM of the access source from which the PBG protection violation is generated. |
| 5 to 1 | Reserved | When read, the value after reset is returned. |
| 0 | WRITE | This bit is set to 1 when an access that has generated the PBG protection violation is the write. |

# Section 34 Data CRC (DCRA)

This section contains a generic description of the data CRC function A (DCRA).

The first part in this section describes the RH850/F1K specific properties, such as the number of units, register base addresses, etc.

The remainder of the section describes the functions and registers of the DCRA.

## 34.1 Features of RH850/F1K DCRA

### 34.1.1 Number of Units

This microcontroller has the following number of DCRA units.

**Table 34.1 Number of Units**

| Product Name | RH850/F1K 100 pins | RH850/F1K 144 pins | RH850/F1K 176 pins |
|---|---|---|---|
| Number of Units | 4 | | |
| Name | DCRAn (n = 0 to 3) | | |

**Table 34.2 Index**

| Index | Description |
|---|---|
| n | Throughout this section, the individual data CRC function A units are identified by the index “n” (n = 0 to 3); for example, DCRAnCTL indicates the DCRAn control register. |

### 34.1.2 Register Base Address

DCRAn base addresses are listed in the following table.

DCRAn register addresses are given as offsets from the base addresses.

**Table 34.3 Register Base Addresses**

| Base Address Name | Base Address |
|---|---|
| <DCRA0_base> | FFF7 $0000_H$ |
| <DCRA1_base> | FFF7 $1000_H$ |
| <DCRA2_base> | FFF7 $2000_H$ |
| <DCRA3_base> | FFF7 $3000_H$ |

### 34.1.3 Clock Supply

The DCRAn clock supply is shown in the following table.

**Table 34.4 Clock Supply**

| Unit Name | Unit Clock Name | Clock Supply Name | Description |
|---|---|---|---|
| DCRAn | PCLK | CPUCLK4 | Module clock |
| | Register access clock | CPUCLK2 | Bus clock |
| | | CPUCLK4 | |

### 34.1.4 Reset Sources

DCRAn reset sources are listed in the following table. DCRAn is initialized by these reset sources.

**Table 34.5 Reset Sources**

| Unit Name | Reset Source |
|---|---|
| DCRAn | All reset sources (ISORES) |

## 34.2 Overview

### 34.2.1 Functional Overview

The data CRC function A can be used to verify or generate CRC protected data streams of arbitrary length and different bit widths.

- 32-bit Ethernet CRC
  $(X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X^{1}+1)$
- 16-bit CCITT CRC
  $(X^{16}+X^{12}+X^{5}+1)$
- CRC of an arbitrary data block length can be generated.
- After initialization of the CRC data register, every write access to the CRC input register generates a new CRC according to the selected polynomial, and the result is stored in the CRC data register.

### 34.2.2 Block Diagram

The following picture shows the block diagram of the data CRC function A.

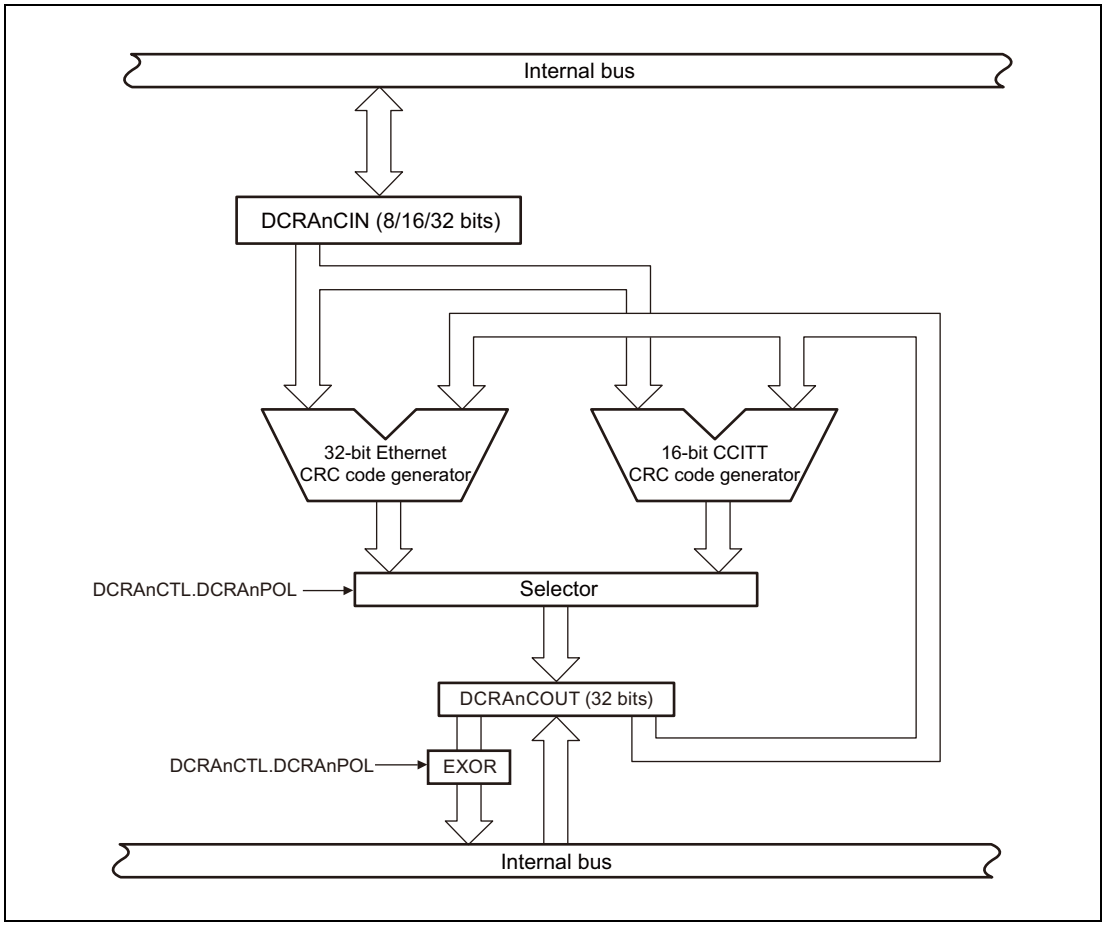
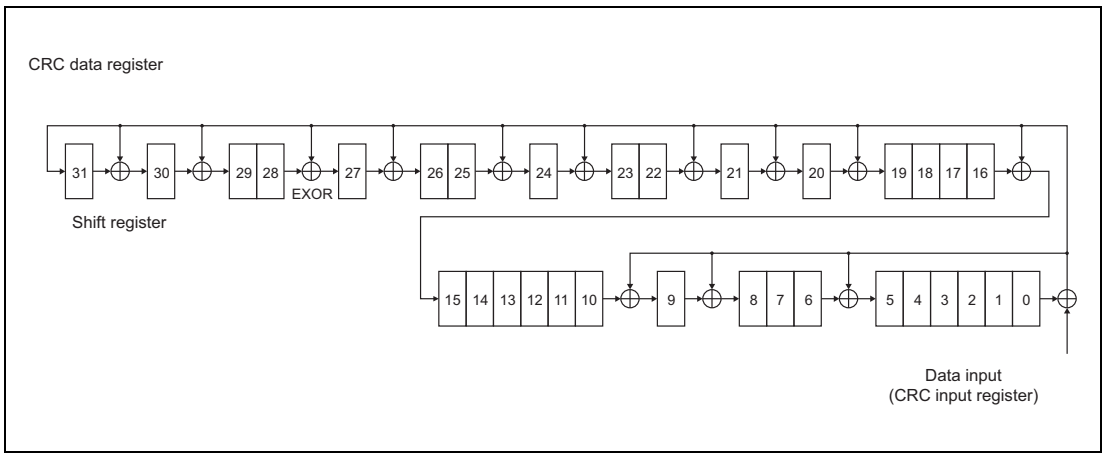


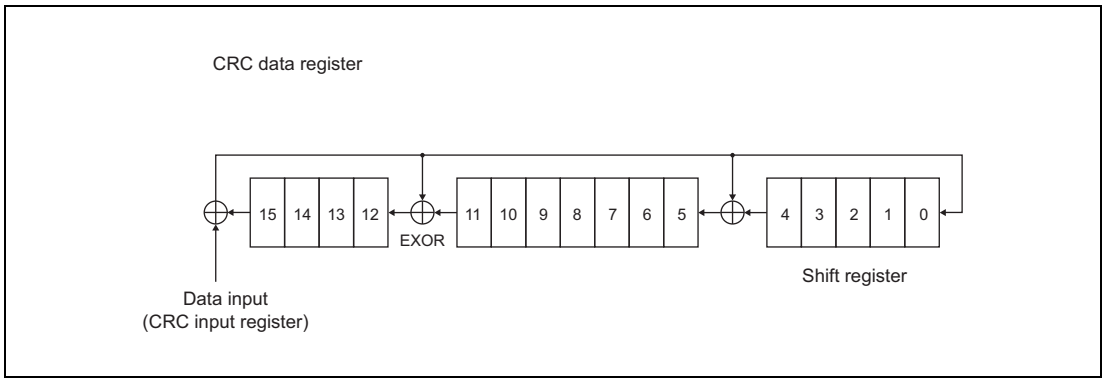

**Figure 34.1 Block Diagram of Data CRC Function A**

### 34.2.3 Operational Circuit

- 32-bit Ethernet




- 16-bit CCITT



## 34.3 Registers

### 34.3.1 List of Registers

DCRA registers are listed in the following table.

For details about <DCRAn_base>, see **Section 34.1.2, Register Base Address**.

**Table 34.6 List of Registers**

| Module Name | Register Name | Symbol | Address |
|---|---|---|---|
| DCRAn | CRC input register | DCRAnCIN | <DCRAn_base> + $00_H$ |
| | CRC data register | DCRAnCOUT | <DCRAn_base> + $04_H$ |
| | CRC control register | DCRAnCTL | <DCRAn_base> + $20_H$ |

### 34.3.2 DCRAnCIN — CRC Input Register

This register holds the input data for CRC calculation. The effective bit width used for CRC calculation must be set by DCRAnCTL.DCRAnISZ[1:0].

When data is written to this register, the CRC code is generated.

The CRC calculation is immediately started after the DCRAnCIN register is written. The DCRAnCOUT register must be initialized with the initial starting value, before the first data of the data block is written to DCRAnCIN register.

**Access:** This register can be read or written in 32-bit units.

**Address:** <DCRAn_base>

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DCRAnCIN[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DCRAnCIN[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 34.7 DCRAnCIN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DCRAnCIN [31:0] | Input Data for CRC Calculation<br>The valid bits are:<br>• For 32 bit effective bit width: DCRAnCIN[31:0]<br>• For 16 bit effective bit width: DCRAnCIN[15:0]<br>• For 8 bit effective bit width: DCRAnCIN[7:0] |

### 34.3.3 DCRAnCOUT — CRC Data Register

This register stores the result of the CRC code generated by the 32-bit Ethernet polynomial or the 16-bit CCITT polynomial.

**Access:** This register can be read or written in 32-bit units.

**Address:** <DCRAn_base> + $4_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DCRAnCOUT[31:16] | | | | | | | | | | | | | | | |
| Value after reset *1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DCRAnCOUT[15:0] | | | | | | | | | | | | | | | |
| Value after reset *1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. The read value after reset is 0000 $0000_H$ since the 32-bit Ethernet CRC polynomial is selected as the CRC generating function after reset.

**Table 34.8 DCRAnCOUT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DCRAnCOUT [31:0] | Result of the CRC Code Generation<br>When the 16-bit CCITT polynomial is enabled, the bits 15 to 0 show the CRC result. The bits 31 to 16 are undefined.<br>The read value of this register is a value obtained by performing EXOR calculation for the following value:<br>• For 32-bit Ethernet polynomial: FFFF $FFFF_H$<br>• For 16-bit CCITT polynomial: $0000_H$<br>For example, when DCRAnCOUT = 5555 $5555_H$ for the 32-bit Ethernet polynomial, AAAA $AAAA_H$ is read. |

**CAUTION**

This register must initialized by setting the initial start value before the first data of the data block is written to DCRAnCIN register.

### 34.3.4 DCRAnCTL — CRC Control Register

This register controls the CRC generation process.

**Access:** This register can be read or written in 8-bit units.

**Address:** <DCRAn_base> + $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | DCRAnISZ[1:0] | | DCRAnPOL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W |

**Table 34.9 DCRAnCTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 3 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 2, 1 | DCRAnISZ[1:0] | Specify the CRC input bit width.<br>00: 32 bits (DCRAnCIN[31:0])<br>01: 16 bits (DCRAnCIN[15:0])<br>10: 8 bits (DCRAnCIN[7:0])<br>11: Setting prohibited |
| 0 | DCRAnPOL | Specifies the CRC generation method.<br>0: 32-bit Ethernet CRC polynomial generation.<br>The byte order of the DCRAnCIN register is LSB (least significant bit) first. This means that, if the CRC input bit width is 8 bits (DCRAnISZ[1:0] = $10_B$), bit positions 7 to 0 of the DCRAnCIN register contain the input data and bit position 0 (LSB) is the start bit of the input data.<br>1: 16-bit CCITT CRC polynomial generation.<br>The byte order of the DCRAnCIN register is MSB (most significant bit) first. This means that, if the CRC input bit width is 8 bits (DCRAnISZ[1:0] = $10_B$), bit positions 7 to 0 of the DCRAnCIN register contain the input data and bit position 7 (MSB) is the start bit of the input data. |

**CAUTION**

- If the CRC generation method (DCRAnCTL.DCRAnPOL) is changed, the DCRAnCOUT register must be initialized by setting the initial start value.
- The CRC input bit width (DCRAnCTL.DCRAnISZ[1:0]) must be set according to the data block bit width. Changing the CRC input bit width is not allowed during processing of a data block (a data block consists of N bytes, half-words or one word). After the final CRC result is read from DCRAnCOUT register, the bit width can be changed. In that case, the DCRAnCOUT register must be initialized with the initial start value.

## 34.4 Operation

The data CRC function A generates a CRC (cyclic redundancy check) of an arbitrary data block length. The data is forwarded to the data CRC function in 8-, 16- or 32-bit units. The CRC polynomial can either be selected for 32-bit Ethernet or 16-bit CCITT. The initial starting value must be set at the DCRAnCOUT register before the first write access to the CRC input register (DCRAnCIN) is performed.

The flowchart below shows the CRC generating procedure.

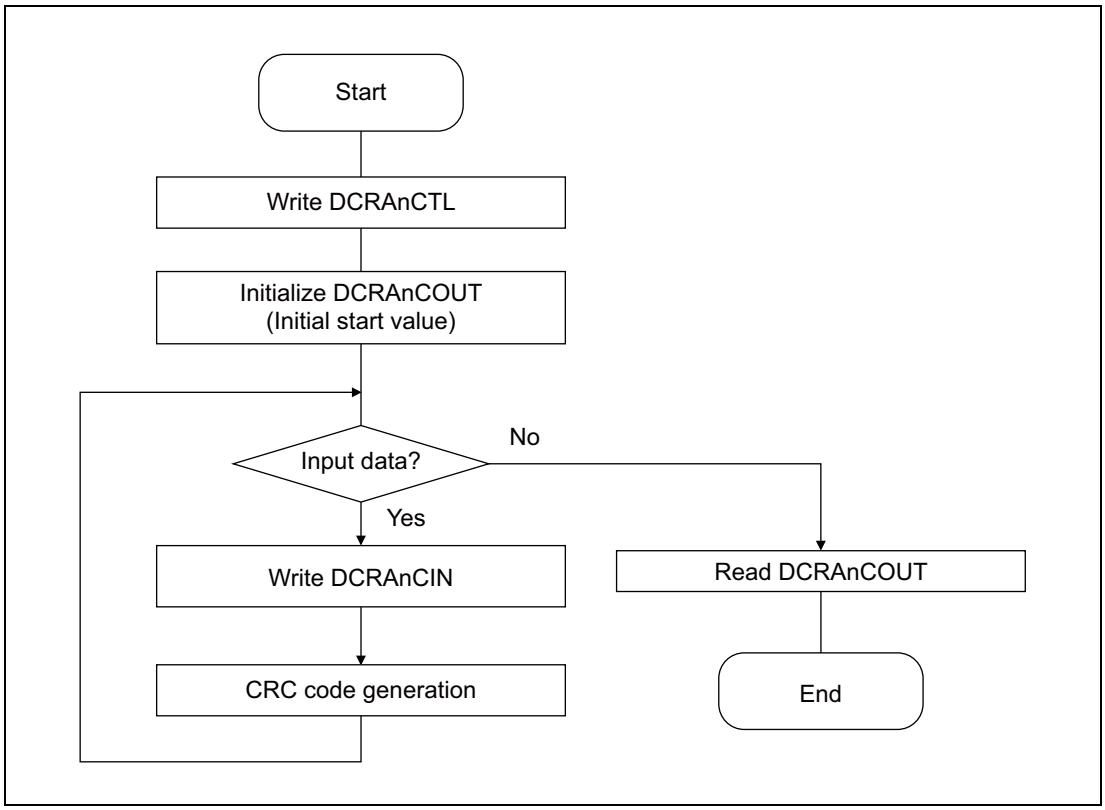



**Figure 34.2 Flowchart of Data CRC Function A**

**NOTES**

1. Before writing the first data to DCRAnCIN, the CRC output register DCRAnCOUT must be initialized with the initial start value.
2. DCRAnCOUT must be re-initialized by setting the initial start value when the polynomial is changed by changing DCRAnCTL.DCRAnPOL.
3. Setting example of the initial start values of the respective polynomials
   The following is the example of setting values.

**Table 34.10 Setting Example of Initial Start Values (When Read at a Reset)**

| | Initial Start Value | EXOR Value | DCRAnCOUT Read Value |
|---|---|---|---|
| 16-bit CCITT | XXXX $FFFF_H$ | XXXX $0000_H$ | XXXX $FFFF_H$ |
| 32-bit Ethernet | $FFFF\ FFFF_H$ | $FFFF\ FFFF_H$ | $0000\ 0000_H$ |

**Note:** X: undefined

# Section 35 Security Function

Please refer to separate volumes about security function which consist of 3 volumes as shown below.

1. Basic Security
2. Secure Watchdog Timer
3. ICUSE

# Section 36 On-Chip Debug Unit (OCD)

This microcontroller has an on-chip debug function. By using the on-chip debug emulator, programs can be debugged with the microcontroller mounted in the target system.

The debug functions incorporated in this microcontroller conform to IEEE-ISTO 5001™-2003 Class 3*1, a Nexus debug interface standard.

**Note 1.** This function is supported only by products with an $\overline{\text{EVTO}}$ pin.

**CAUTION**

**The debug functions described in this section are supported by the microcontroller but whether they are usable depends on the debugger. For details on debugging, see the user's manual of the debugger.**

## 36.1 Overview of RH850/F1K OCD

### 36.1.1 Functional Overview

The on-chip debug functions described below are supported by the microcontroller.

#### (1) Debug interface

This microcontroller supports the following as debug interfaces: Nexus Interface, Low Pin Debug Interface (1-pin) - hereinafter called “LPD (1-pin)”, and Low Pin Debug Interface (4-pin) - hereinafter called “LPD (4-pin)”.

On-chip debug can be performed using these debug interfaces.

**NOTE**

When LPD (1-pin) is used, LPD is operating by the clock of MainOSC.

#### (2) Debug monitoring function

Debug-dedicated monitor program space is mounted and is used during debugging.

The basic debug functions below can be used by running a monitoring program.

- Downloading the user-created program
- Reading and writing the memory and registers
- Running the user-created program starting at any address

#### (3) On-chip break

A maximum of 12 breakpoints can be specified at any execution address. Of the 12 breakpoints, a maximum of four breakpoints can be specified for any access (access address, access data).

#### (4) Software break

Software break points can be specified at any execution address.

#### (5) Peripheral break

The peripheral break function generates a stop request to the peripheral modules of the microcontroller if the user-created program is stopped, for instance upon a breakpoint hit.

#### (6) Forced break

Execution of the user-created program can be interrupted forcibly.

#### (7) Forced reset

This device (microcontroller) can be forcibly reset.

#### (8) Real time RAM monitoring (RRM)

The memory can be read during program execution. Because this read access uses debug-dedicated DMA, it has minimal effect on program execution.

#### (9) Dynamic memory modification (DMM)

The memory can be written during program execution. Because this write access uses debug-dedicated DMA, it has minimal effect on program execution.

#### (10) Timer function

Using a 32-bit counter, the time for running the user-created program can be measured based on the clock for debug.

For the measurement accuracy, see the user's manual of the debugger.

#### (11) Mask function

Masking the following factors is possible.

- All reset sources except for a POC reset and a wakeup reset

#### (12) Hot plug-in function

Debugging can be started in normal operating mode without external reset input.

**NOTE**

When the hot plug-in function is used in power save mode, the INTDCUTDI interrupt is required to return from power save mode as the wake-up process. About the INTDCUTDI interrupt, see Table 7.4, EI Level Maskable Interrupt Sources (Channel No. 21).

#### (13) Security function

To prevent the contents of the flash memory from being read by an unauthorized person, a 128-bit ID code can be written to the microcontroller. If the code the user inputs when starting a debugger does not match the ID code written to the microcontroller, the flash memory cannot be accessed.

For details on how to set the ID code, see the user's manual of the debugger.

#### (14) Trace function

Execution history, data changes, etc. of the user-created program can be obtained.

**NOTE**

The trace function is only available in devices with 2-MB memory.

### 36.1.2 External Input/Output Pins

Table 36.1 External Input/Output Pins

| Pin | Special Function | PKG No | | |
|---|---|---|---|---|
| | | 100 pins | 144 pins | 176 pins |
| JP0_0 | DCUTDI/LPDI/LPDIO | √ | √ | √ |
| JP0_1 | DCUTDO/LPDO | √ | √ | √ |
| JP0_2 | DCUTCK/LPDCLK | √ | √ | √ |
| JP0_3 | DCUTMS | √ | √ | √ |
| JP0_4 | $\overline{\text{DCUTRST}}$ | √ | √ | √ |
| JP0_5 | $\overline{\text{DCURDY}}$/LPDCLKOUT | √ | √ | √ |
| JP0_6 | $\overline{\text{EVTO}}$ | | √ | √ |

## 36.2 Peripheral Break Control

The peripheral break function generates a stop request to the peripheral modules of the microcontroller if the user-created program is stopped, for instance upon a breakpoint hit.

During peripheral break, the peripheral modules operate as follows.

### a. Modules that stop unconditionally regardless of the EPC.SVSTOP setting

Table 36.2 Modules that Stop Unconditionally Regardless of the EPC.SVSTOP Setting

| Module |
|---|
| Window watchdog timer (WDTA) |

### b. Modules that continue to operate by the setting of emulation registers even when EPC.SVSTOP = 1

Table 36.3 Modules that Continue to Operate by the Setting of Emulation Registers even when EPC.SVSTOP = 1

| Module | Emulation Register | n |
|---|---|---|
| OS timer (OSTM) | OSTMnEMU.OSTMnSVSDIS<br>0: Stops during break<br>1: Continues during break | 0 to 4 |
| Timer array unit D (TAUD) | TAUDnEMU.TAUDnSVSDIS<br>0: Stops during break<br>1: Continues during break | 0 |
| Timer Array Unit B (TAUB) | TAUBnEMU.TAUBnSVSDIS<br>0: Stops during break<br>1: Continues during break | 0, 1 |
| Timer array unit J (TAUJ) | TAUJnEMU.TAUJnSVSDIS<br>0: Stops during break<br>1: Continues during break | 0, 1 |
| Real-Time Clock (RTCA) | RTCAnEMU.RTCAnSVSDIS<br>0: Stops during break<br>1: Continues during break | 0 |
| Clocked serial interface G (CSIG) | CSIGnEMU.CSIGnSVSDIS<br>0: Stops during break<br>1: Continues during break | 0, 1 |
| Clocked serial interface H (CSIH) | CSIHnEMU.CSIHnSVSDIS<br>0: Stops during break<br>1: Continues during break | 0 to 3 |
| Timer Motor Control Function (TAPA) | TAPAnEMU.TAPAnSVSDIS<br>0: Stops during break<br>1: Continues during break | 0 |
| Encoder Timer (ENCA) | ENCAnEMU.ENCAnSVSDIS<br>0: Stops during break<br>1: Continues during break | 0 |
| PWM Output/Diagnostic (PWM-Diag) | PWBAnEMU.PWBAnSVSDIS<br>0: Stops during break<br>1: Continues during break | 0 |
| | PWSAnEMU.PWSAnSVSDIS<br>0: Stops during break<br>1: Continues during break | 0 |
| A/D converter (ADCA) | ADCAnEMU.ADCAnSVSDIS<br>0: Stops during break<br>1: Continues during break | 0, 1 |

**CAUTION**

**For details on the registers, see the register description of the corresponding section.**

### c. Modules that stop when EPC.SVSTOP = 1

Table 36.4 Modules that Stop when EPC.SVSTOP = 1

| Module |
|---|
| LIN/UART interface (RLIN3) |
| Low-Power Sampler (LPS) |

## 36.3 Hot Plug-in in Each Mode

### 36.3.1 RUN Mode

When Hot Plug-in has occurred in RUN mode, it is necessary to maintain the current state.

### 36.3.2 STOP/DeepSTOP Mode

When Hot Plug-in has occurred in STOP/DeepSTOP mode, it is necessary to transfer in RUN mode.

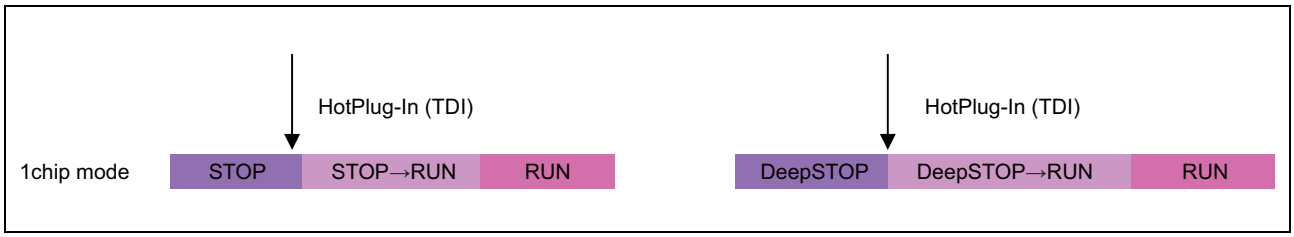



Figure 36.1 Hot Plug-in in STOP/DeepSTOP mode

### 36.3.3 Cyclic RUN Mode

When Hot Plug-in has occurred in Cyclic RUN mode, it is necessary to transfer in DeepSTOP mode and wakeup to RUN mode.

All regulators in ISO are made operating state in OCD mode.

Then, on the occurrence of interrupt of “DCUTDI Low level Detection interrupt (INTDCUTDI)”, it is necessary for a transfer order to be executed to DeepSTOP mode by user (software) if on-chip debugging is performed.

The sequence of the time when to enter to OCD mode in Cyclic RUN mode is mentioned below.

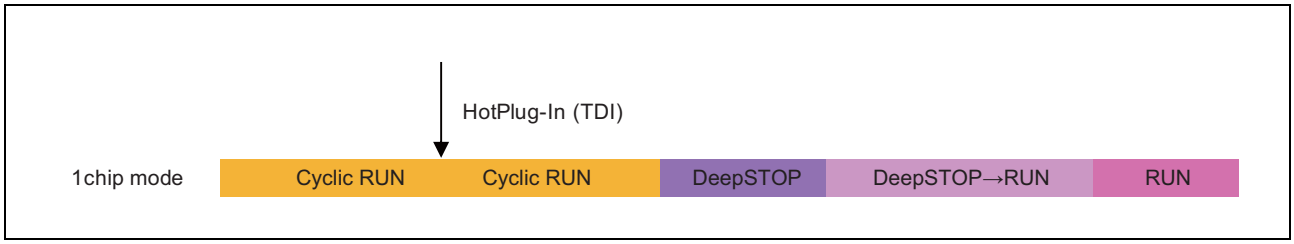



Figure 36.2 Hot Plug-in in Cyclic RUN mode

### 36.3.4 Cyclic STOP Mode

When Hot Plug-in has occurred in Cyclic STOP mode, it is necessary to transfer in Cyclic RUN mode.

Then, on the occurrence of wakeup factor of DCUTDI, it is necessary for a transfer to be executed as the same case of Cyclic RUN mode.

The sequence of the time when to enter to OCD mode in Cyclic STOP mode is mentioned below.

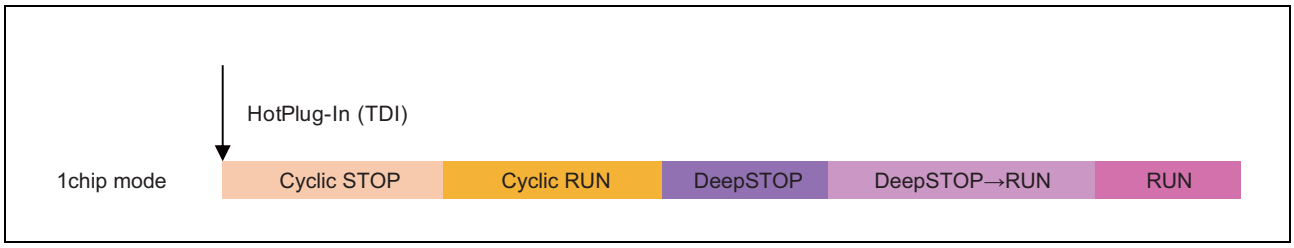



**Figure 36.3 Hot Plug-in in Cyclic STOP mode**

# 36.4 Registers

## 36.4.1 EPC — Emulation Peripheral Control Register

This register stops operation of peripheral functions (timer, serial interface, and A/D converter) in debug mode (SVSTOP).

**Access:** Accessing from the user program is prohibited.

**Address:** —

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | SVSTOP | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | — | — | — | — | — | — | — | — |

**Table 36.5 EPC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | — |
| 6 | SVSTOP | Stops operation of peripheral functions (timer, serial interface, and A/D converter) during debugging.<br>0: Does not stop operation<br>1: Stops operation |
| 5 to 0 | Reserved | — |

**NOTE**

EPC is set by the debugger. Setting by the user program is prohibited. As for the setting of the debugger, see the user's manual of the debugger.

## 36.5 Cautions on Using On-Chip Debugging

### 36.5.1 Treatment of Devices Used for Debugging

Do not install a device that was used for debugging on a mass-produced product, because the flash memory was rewritten during system debugging and thus the write/erase endurance of the flash memory cannot be guaranteed.

### 36.5.2 Reset Assertion When a Debugger is Connected

If a program in which a reset is asserted at the start of program execution is executed when a debugger is being used, the microcontroller is reset before preparation for communications between the OCD emulator and microcontroller is complete. For this reason, communications may not proceed correctly.

The period of preparation for communications depends on the host PC environment of the OCD emulator and the operating frequency of the microcomputer. To ensure that the debugger operates properly when debugging a program in which a reset is asserted at the start of program execution, insert a wait between the start of program and reset assertion.

### 36.5.3 Restrictions When HS IntOSC is Used as the Main Clock Source Instead of MainOSC

Need to change the debug interface configuration (OPJTAG) to LPD (4-pin) in advance if E1 Emulator is used for debug.

LPD (1-pin) mode can’t be used.

### 36.5.4 Restrictions When the Writing of OCD_MD and RESET are Occur at the Same Time, or Restrictions When the Writing of MTR (DBG_CTRLP) and RESET are Occur at the Same Time

When the writing of OCD_MD and RESET are occur at the same time, or the writing of MTR (DBG_CTRLP) and RESET are occur at the same time, it is a possibility of terminal hazard.

If the debugger was disconnected, please reconnect the debugger.

### 36.5.5 Transition to DeepSTOP Mode When a Debugger is Connected

When a debugger is in use, when a program to transition to DeepSTOP mode is executed immediately following the start of the program, the microcomputer stops power supply to the Isolated area before preparation for communications between the OCD emulator and microcomputer is complete. For this reason, communications may not proceed correctly.

The period of preparation for communications depends on the host PC environment of the OCD emulator and the operating frequency of the microcomputer. A wait must be inserted during the period from release from the reset state and transition to DeepSTOP mode by the emulator for correct operation of the debugger when debugging a program to transition to DeepSTOP mode immediately after the program starts.

# Section 37 Flash Memory

This section describes the flash memory mounted on RH850/F1K.

The first part in this section describes the characteristics of the mounted flash memory and the characteristics specific to RH850/F1K, such as the memory map, flash memory programming, and ECC.

## 37.1 Features

- Includes code flash memory and data flash memory
  The code flash memory can store program codes and data and has the user area and the extended user area.
  The data flash memory is used for storing data.
- Method of flash memory programming
  Flash memory programming via a serial interface and programming of flash memory by a user program (self-programming) are supported.
- Support for BGO (Back Ground Operation)
  The BGO function allows programs to be executed in the code flash memory while the data flash memory is being programmed/erased.
- Flash memory data security
  - Support for security functions to protect against illicit tampering with or reading out of data in the flash memory
  - Support for protection functions to protect against erroneous overwriting of the flash memory
- Option byte function
  Sets the operation after releasing reset for ports, WDTA, CVM, and CAN FD CRC(Supported only for PREMIUM), and RESETOUT.
- Support for the error detection/correction function (ECC) in the code flash memory and data flash memory
  Built-in ECC function can detect 2-bit errors and detect/correct 1-bit errors.
- Interrupts can be acknowledged in self-programming mode.

For code flash sizes and data flash sizes of each product, see the following sections.

- **Section 4.1, Address Space**

## 37.2 Structure of Memory

### 37.2.1 Mapping of Code Flash Memory

**Figure 37.1** illustrates the mapping of the code flash memory for the 2 MB, 1.5 MB, 1 MB, and 768 KB devices. The user area of the code flash memory of the RH850/F1K is divided into 8 and 32 KB blocks, which serve as the units of erasure. A single 32 KB block is also incorporated as the extended user area. The user area and extended user area are available as areas for storing the user program.

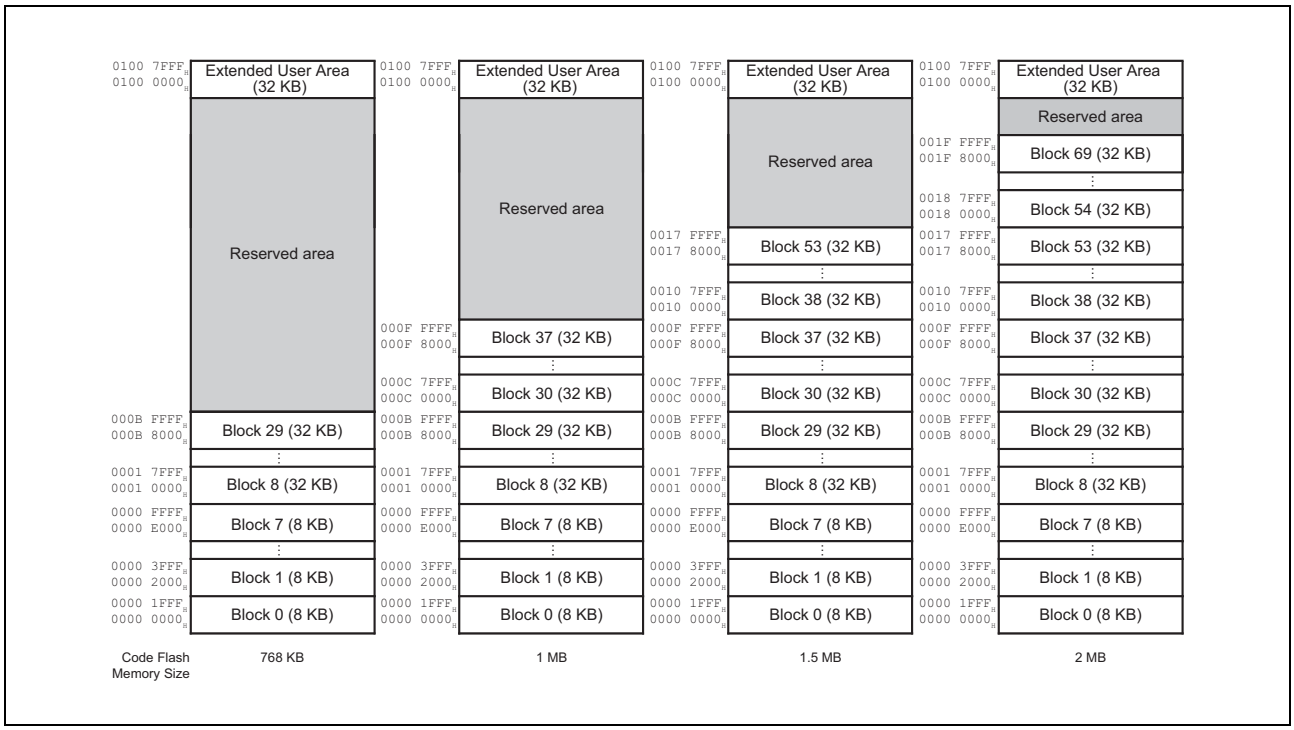



**Figure 37.1 Mapping of the Code Flash Memory**

### 37.2.2 Mapping of Data Flash Memory

The data area of the data flash memory in the RH850/F1K is divided into 64-byte blocks, with each being a unit for erasure. **Figure 37.2** shows the mapping of the data flash memory.

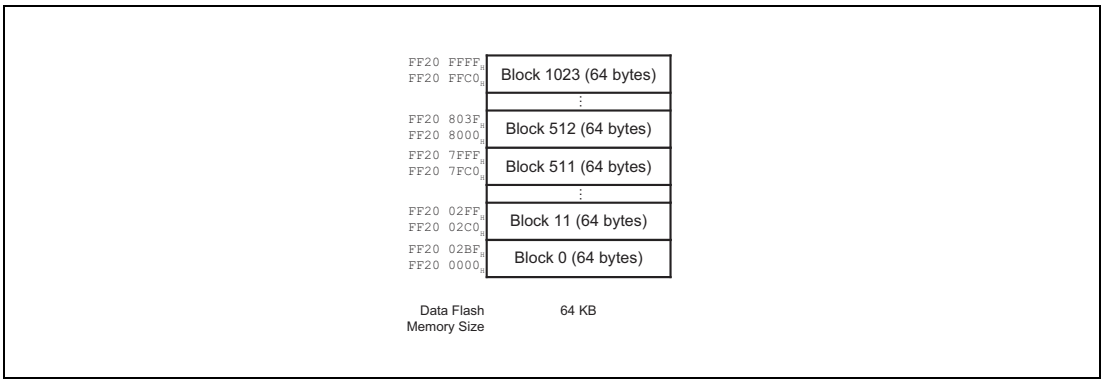



**Figure 37.2 Mapping of the Data Flash Memory**

## 37.3 Operating Modes Associated with Flash Memory

**Figure 37.3** is a diagram of the mode transitions associated with the flash memory. For the procedures for setting the modes, see **Section 6, Operating Mode**.

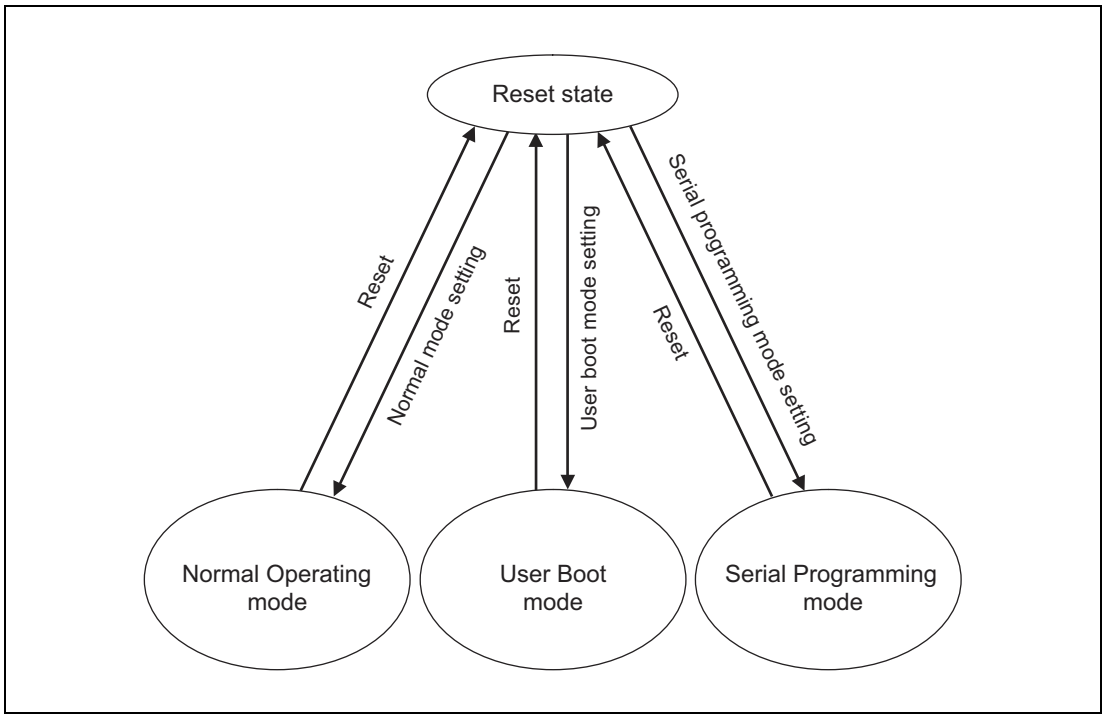



**Figure 37.3 Mode Transition Associated with Flash Memory**

**Table 37.1** shows the flash memory area which is programmable and erasable in each mode and the boot program after reset release.

**Table 37.1 Programmable and Erasable Area in Each Mode and the Boot Program after Reset Release**

| Item | Normal Operating Mode | User Boot Mode | Serial Programming Mode |
|---|---|---|---|
| Programmable and erasable area | • User area<br>• Extended user area<br>• Data area | • User area<br>• Data area | • User area<br>• Extended user area<br>• Data area |
| Boot program after reset release | Program in user area or extended user area (Changeable by using the variable reset vector) | Program in extended user area. (Reset vector is 0100 $0000_H$) | Firmware program for serial programming |

## 37.4 Functions

### 37.4.1 Functional Overview

The flash memory of the RH850/F1K can be updated via a serial interface by a dedicated flash memory programmer (serial programming), before being mounted on the target system or on a flash adapter system.

Furthermore, security functions to prohibit updating of the user program written in the flash memory are incorporated, and this can prevent tampering by third parties.

Programming by the user program (self-programming) is suited for applications where the target system program may require updating after deployed to the end user. Protection features for the safe rewriting of the flash memory are also incorporated. Furthermore, interrupt processing during self-programming is supported, so programming can proceed at the same time as external communications, etc., and this allows programming under various conditions. **Table 37.2** gives an overview of the methods of programming and the corresponding operating modes.

**Table 37.2 Methods of Programming**

| Method of Programming | Description | Operating Mode |
|---|---|---|
| Serial programming | A dedicated flash memory programmer allows on-board programming of the flash memory after the device is mounted on the target system. | Serial programming mode |
| | A dedicated flash memory programmer and dedicated programming adapter board allow off-board programming of the flash memory, i.e. programming of the device before it is mounted on the target system. | |
| Self-programming | The user program that is written to code flash memory in advance by serial programming also allows updating the flash memory.<br>The background operation capability makes it possible to fetch instructions or otherwise read data in code flash memory while the data flash memory is self-programming.<br>For this reason, it is possible to update the data flash memory by executing a program written to the code flash memory.<br>Instructions in the code flash memory cannot be fetched and data cannot be accessed while the code flash memory is being updated by self-programming. In such cases, a program for updating must be transferred to the local RAM in advance and executed. | Normal operating mode<br>User boot mode |

Renesas provides a library for self-programming. For details on this library, see the user's manuals for the code flash library and data flash library of this device.

**Table 37.3** lists the functions of the flash memory. Dedicated flash memory programmer commands enable serial programming, while reading of the flash memory by a library function or the user program enables self-programming.

Table 37.3 Basic Functions at a Glance

| Function | Description | Level of Support (√: Supported, △: Conditionally Supported, —: Not Supported) | |
|---|---|---|---|
| | | Serial programming | Self-programming |
| Blank checking | This is used to check a specified block to ensure that writing to it has not already proceeded. Results of reading from code flash memory and data flash memory to which nothing has been written after erasure are not guaranteed, so use blank checking to confirm that writing to memory has not proceeded after erasure. | √ | △<br>(Only data flash is supported) |
| Block erasure | This is for erasing the contents of a specified block of memory. | √ | √ |
| Programming | This is for writing to a specified address. | √ | √ |
| Verification and checksum | Data that are read out from flash memory are compared with data transferred from the flash memory programmer. | √ | —<br>(Reading of data by the user program is possible) |
| Reading | Data that have been written to the flash memory are read out. | √ | √ |
| Setting for OTP (one-time programming) | A specified block of code flash memory is set for OTP (OTP can only be set, that is, it is not possible to release a block's OTP setting). | √ | √ |
| Setting an ID | An ID setting is made for use in controlling the connection of a dedicated flash memory programmer for serial programming, controlling of the on-chip debugger, and programming of the code flash memory by self-programming. | √ | √ |
| Security settings | Security settings are for use in serial programming. | √ | △<br>(Only when setting is prohibited after being permitted) |
| Protection settings | Settings for block protection of code flash memory and variable reset vector are provided. | △<br>(Setting of the reset vector values for variable reset vector function is not supported.) | √ |
| Setting of option bytes | Option bytes are set to change them from the initial values for the RH850/F1K. | √ | √ |
| Clearing the configuration | ID setting, security settings, protection settings, and option byte settings are initialized. | √ | — |

For details on serial programming, see the user’s manual of the flash programmer.

For details on self-programming, see the user’s manuals for the code flash library and data flash library of this device.

The flash memory supports various security functions.

The OTP setting and authentication of the ID code are security functions for use with serial programming and self-programming.

In serial programming, authentication of the ID code, prohibiting connection of a dedicated flash memory programmer, and prohibition of commands (for block erasure, programming, and reading) are available for use as security functions.

The security functions supported by the flash memory are listed in **Table 37.4**.

For details on security function, see the *RH850/F1K Basic Security User's Manual:Hardware*.

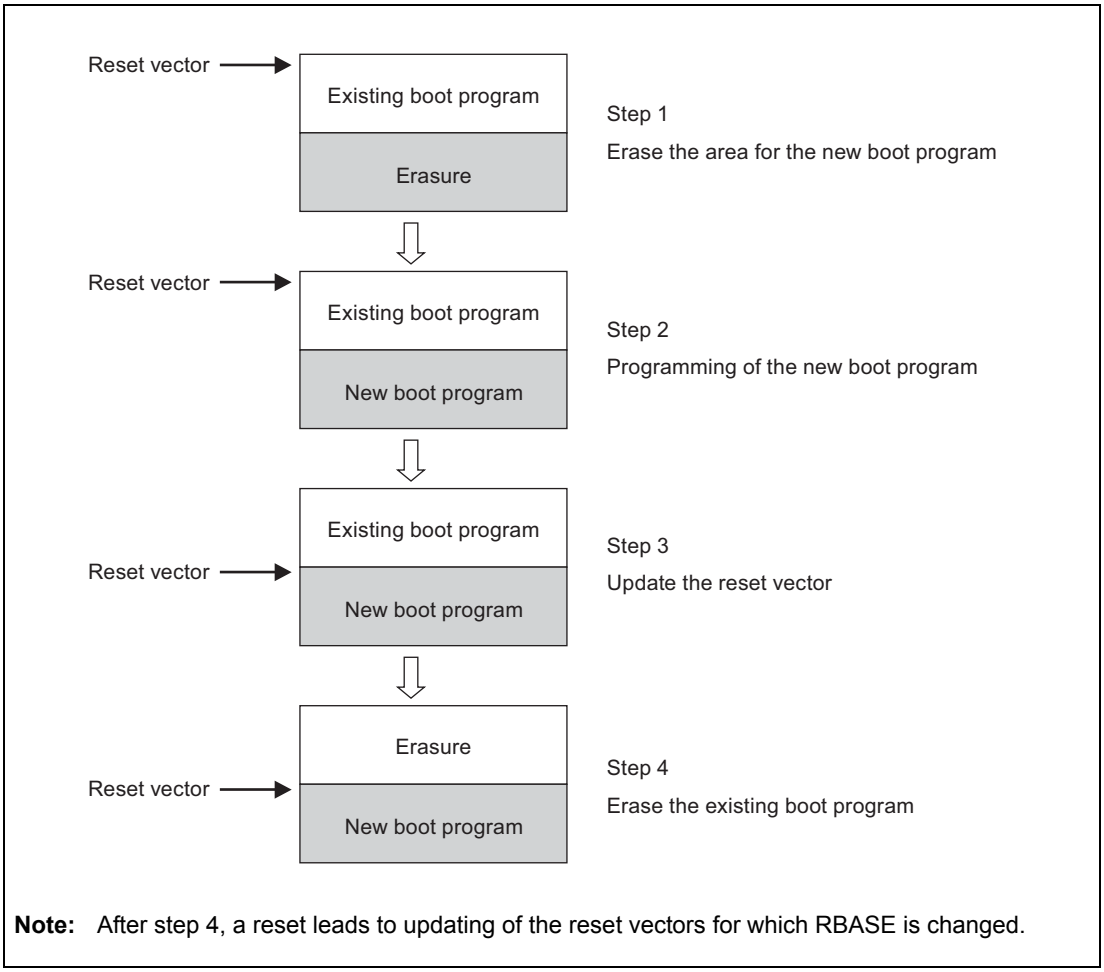
**Table 37.4 Summary of Security Functions**

| Function | Description |
|---|---|
| OTP | OTP can be individually set for each block of the user area and the extended user area of code flash memory. When the OTP setting is made for an area, programming by serial programming and by self-programming is prohibited. Once set, the OTP setting cannot be released. Furthermore, since execution of the configuration clearing command is prohibited for any area for which OTP has been set, changing a security setting from “prohibited” to “permitted” is not possible. |
| ID authentication | The result of ID authentication can be used to control the connection of a dedicated flash memory programmer for serial programming. The result of ID authentication can also be used to control enabling of self-programming. |
| Prohibition of connection of a dedicated flash memory programmer | The connection of a dedicated flash memory programmer for serial programming is prohibited. Since execution of the configuration clearing command is also prohibited when the connection of a dedicated flash memory programmer is prohibited, changing a security setting from “prohibited” to “permitted” is not possible. |
| Prohibition of block erasure commands | Block erasure commands at the time of serial programming are prohibited. Since execution of the configuration clearing command is also prohibited when block erasure commands are prohibited, changing a security setting from “prohibited” to “permitted” is not possible. |
| Prohibition of programming commands | Block erasure commands and programming commands at the time of serial programming are prohibited. Only through execution of the configuration clearing command can the prohibition be lifted. |
| Prohibition of read commands | Read commands at the time of serial programming are prohibited. Only through execution of the configuration clearing command can the prohibition be lifted. |

The flash memory supports various protection functions. The protection functions supported by the flash memory are listed in **Table 37.5**.

**Table 37.5 Summary of Protection Functions**

| Function | Description |
|---|---|
| Block protection | Lock bit settings can be individually made to enable or disable programming and erasure of each block of the user area and the extended user area of code flash memory. Programming and erasure by self-programming of an area for which the lock bit is set and the lock bit function is enabled are prohibited.<br>Programming or erasure can proceed again when the lock bit function is disabled after having been enabled. When a block of code flash memory is erased, the lock bit for that block is also erased. |
| Hardware protection | The level on the FLMD0 pin can be set to prohibit programming and erasure of the code flash memory.<br>- FLMD0 = 0: Programming prohibited<br>- FLMD0 = 1: Programming permitted |
| Variable reset vector | The protection settings include control of the reset vector. As shown in **Figure 37.4**, after programming of a new boot program while leaving the existing boot program in place, changing the reset vector is a safe way to change to the area holding the new boot program.<br>The areas that can be specified by using the reset vector are the user area and extended user area. |




**Note:** After step 4, a reset leads to updating of the reset vectors for which RBASE is changed.

**Figure 37.4 Utilizing the Variable Reset Vector Function to Update the Boot Program**

## 37.5 Serial Programming

A dedicated flash memory programmer can be used to handle flash memory in serial programming mode.

#### Serial programming

The microcontroller is mounted on the system board at the time of serial programming. Providing a connector to the board enables handling of the microcontroller by the flash memory programmer to proceed.

### 37.5.1 Environments for Programming

The recommended environments for handling the flash memory of the microcontroller with data are described below.

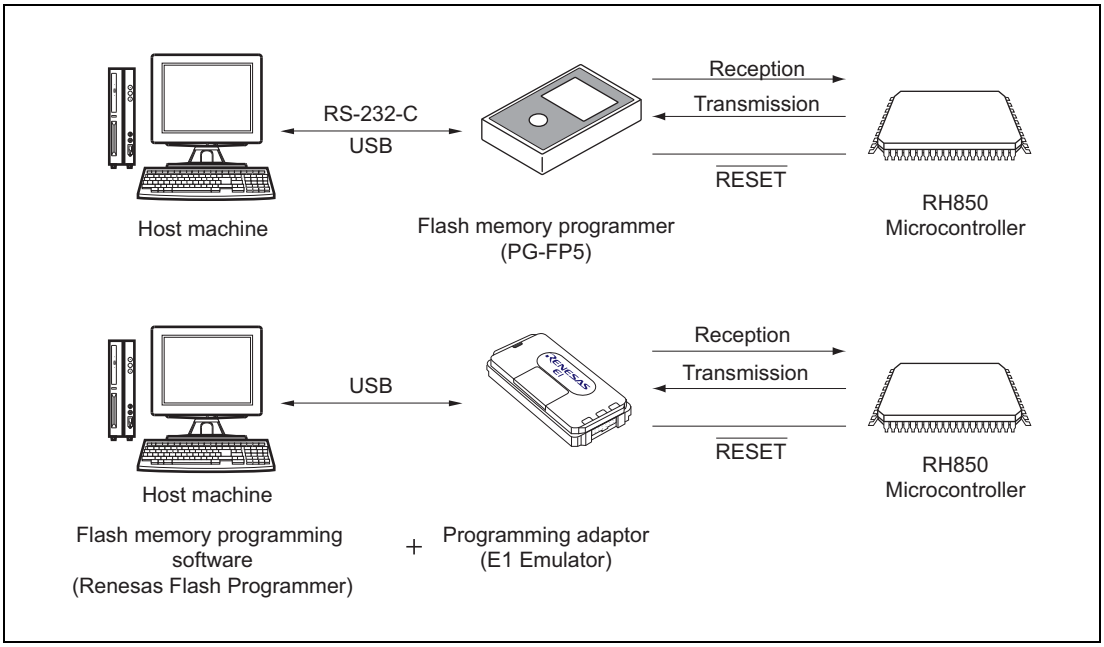



**Figure 37.5 Environments for Handling Programs of the Flash Memory**

By using the PG-FP5 flash memory programmer or the combination of the Renesas Flash Programmer (software for writing to flash memory) running on the host machine and the E1 emulator as an programming adaptor, the user is easily able to erase, program, and verify the contents of the on-chip memory of flash-memory-equipped microcontrollers from Renesas Electronics.

The PG-FP5 flash memory programmer handles programming from a host machine or programming in stand-alone mode while the Renesas Flash Programmer only handles programming from a host machine.

**NOTE**

For details on the PG-FP5, see the *PG-FP5 Flash Memory Programmer User's Manual*. For details on the Renesas Flash Programmer of flash programming software, see the *Renesas Flash Programmer Flash Programming Software User's Manual*.

## 37.6 Communication Modes

### 37.6.1 Asynchronous Flash Programming Interface - 1-Wire UART

The single-wire asynchronous serial programming interface, 1-wire UART is connected to the flash memory programmer with the following port.

- FPDR (JP0_0): Receive data input/transmit data output

### 37.6.2 Asynchronous Flash Programming Interface - 2-Wire UART

The double-wire asynchronous serial programming interface, 2-wire UART is connected to the flash memory programmer with the following ports.

- FPDR (JP0_0): Receive data input
- FPDT (JP0_1): Transmit data output

### 37.6.3 Synchronous Flash Programming Interface CSI

The synchronous serial programming interface CSI is connected to the flash memory programmer with the following ports.

- FPDR (JP0_0): Receive data input
- FPDT (JP0_1): Transmit data output
- FPCK (JP0_2): Serial clock input

The flash memory programmer outputs the serial data clock SCK, and the microcontroller operates as a slave.

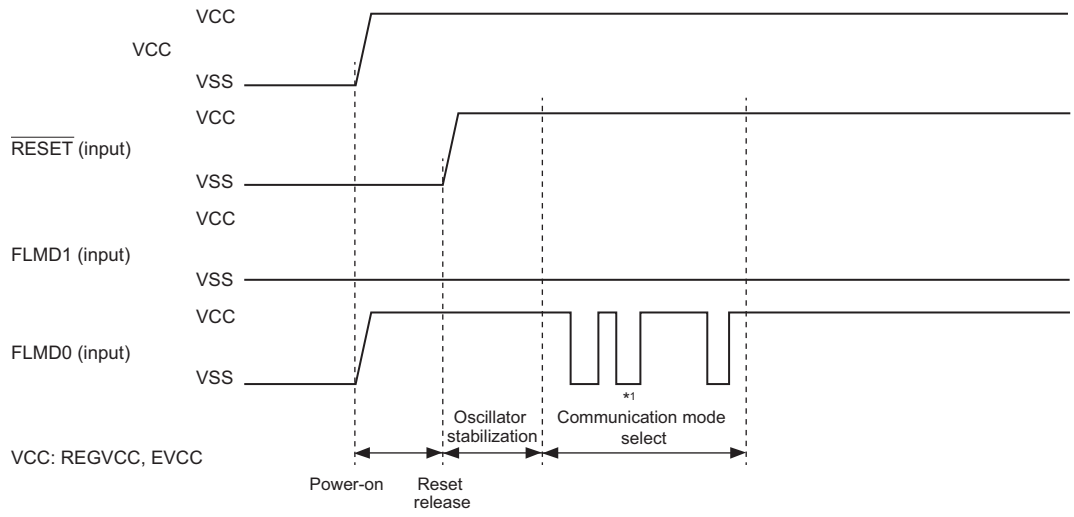
**NOTE**

For details on Renesas Flash Programmer, see the *Renesas Flash Programmer Flash Programming Software User's Manual*.

### 37.6.4 Selection of Communication Method

In RH850/F1K, communication method can be selected by pulse input to the FLMD0 pin (up to 7 pulses) after transition to the flash memory programming mode. The FLMD0 pulse is generated by a dedicated flash memory programmer.

**Figure 37.6** shows the relation between the number of pulses and communication method.




Note 1. The number of clock cycles to be inserted depends on the communication method as in the following table.

| FLMD0 Pulse | Communication Method |
|---|---|
| 0 | 2-wire UART |
| 7 | 1-wire UART |
| 3 | CSI |
| Other than above | Setting prohibited |

**Figure 37.6 Selection of Communication Method**

# 37.7 Self-Programming

## 37.7.1 Outline

The RH850/F1K supports programming of the flash memory by the user program itself. Renesas Electronics provides a code flash library and a data flash library for use with user programs. These libraries can be used for writing to the code flash memory and to the data flash memory.

When the data flash memory is programmed, the background operation facility makes it possible to execute a programming program from the code flash memory to program the data flash memory. Furthermore, the programming program can be copied to local RAM in advance of the programming operation, and executed from the given destination to perform the programming.

The programming program can be copied to the local RAM in advance and executed to program the code flash memory.

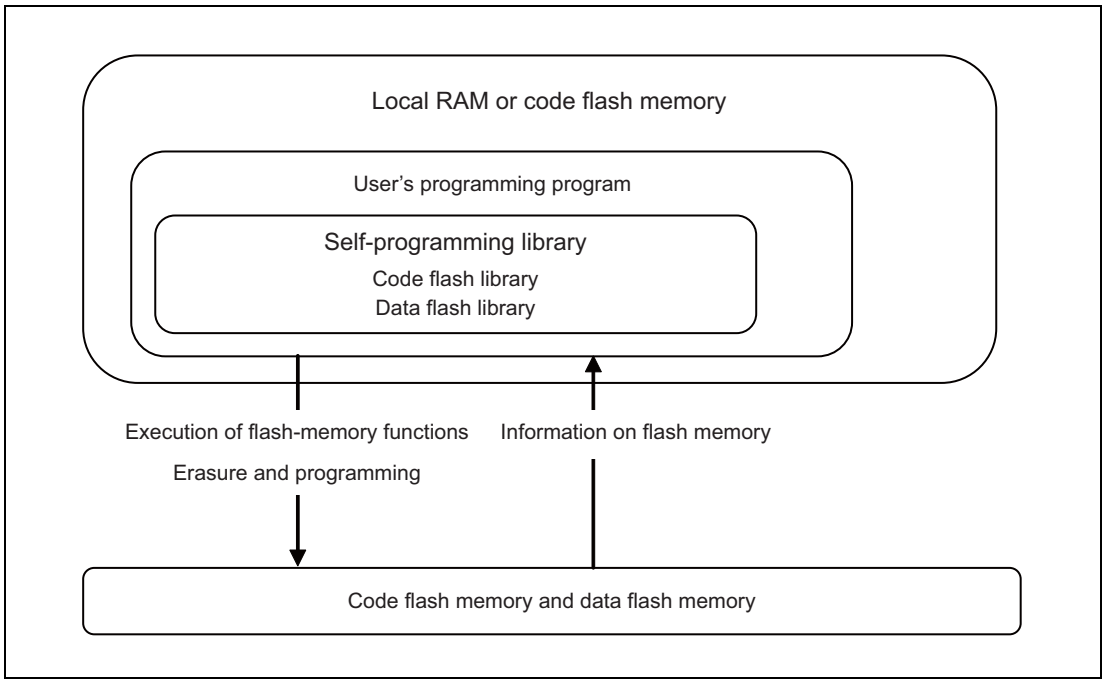



**Figure 37.7 Schematic View of Self-Programming**

For details on the self-programming of flash memory, see the user's manuals for the code flash and data flash libraries for this device.

### 37.7.2 Background Operation

Background operations can be used when the combination of the flash memory for writing and the flash memory for reading is any of those listed below.

**Table 37.6 Conditions under which Background Operation is Usable**

| Range for Writing | Range for Reading |
|---|---|
| Data flash memory | Code flash memory |

### 37.7.3 Enabling Self-Programming

The self-programming function can be activated in normal operating mode and user boot mode.

Erasure and programming of the code flash memory by the self-programming function is enabled by making the FLMD0 pin high level.

This prevents unnecessary overwriting of the program if the device operates incorrectly.

The FLMD0 pin is made high level by using one of the following methods.

- The FLMD0 pin is externally pulled up.
- The FLMD0 pin is pulled up by the FLMDCNT register.

The outline of the FLMDCNT register is described in **Section 37.7.3.1, FLMDCNT Register**.

#### 37.7.3.1 FLMDCNT Register

This register specifies the internal pull-up or pull-down of the FLMD0 pin.

The correct write sequence using the FLMDPCMD register is required in order to update this register. For details, see **Section 5, Write-Protected Registers**.

**Access:** This register can be read or written in 32-bit units.

**Address:** FFA0 0000$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | FLMDPUP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 37.7 FLMDCNT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 0 | FLMDPUP | FLMD0 Pin Software Control<br>0: Pull-down selected<br>1: Pull-up selected |

## 37.8 Reading Flash Memory

### 37.8.1 Reading Code Flash Memory

Special settings are not required to read code flash memory in normal mode. Data can simply be read out through access to addresses in the code flash memory.

Reading from an area of code flash memory that has been erased but not yet been programmed again (i.e. that is in the non-programmed state) can lead to the detection of an ECC error and generation of the corresponding exception.

### 37.8.2 Reading Data Flash Memory

Configure the number of read cycles in the EEPRDCYCL register prior to reading data from data flash memory in normal mode. Once this register is properly configured, data can be read by simply accessing addresses in the data flash memory.

Values read from data flash memory that has been erased but not yet been programmed again are undefined. Use blank checking when you need to confirm that an area is in the non-programmed state.

### 37.8.2.1 EEPRDCYCL — Data Flash Wait Cycle Control Register

This register is used to specify the number of wait cycles to be inserted when reading the data in the data flash.

Set the number of wait cycles to be inserted in the clock cycle when reading the data flash according to the operating clock frequency of the CPU ($f_{CPUCLK}$).

**Access:** This register can be read or written in 8-bit units.

**Address:** FFC5 $9810_H$

**Value after reset:** $0F_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | WAIT[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 37.8 EEPRDCYCL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned. When writing, write the value after reset. |
| 3 to 0 | WAIT[3:0] | Number of Wait Cycles |

| WAIT [3:0] | Number of Wait Cycles | CPU Operation Frequency | | |
|---|---|---|---|---|
| | | $f_{CPUCLK}$ ≤ 40 MHz | 40 MHz < $f_{CPUCLK}$ ≤ 80 MHz | 80 MHz < $f_{CPUCLK}$ ≤ 120 MHz |
| 0000 | 1 | √ | Setting prohibited | Setting prohibited |
| 0001 | 2 | √ | √ | Setting prohibited |
| 0010 | 3 | √ | √ | √ |
| 0011 | 4 | √ | √ | √ |
| 0100 | 5 | √ | √ | √ |
| 0101 | 6 | √ | √ | √ |
| 0110 | 7 | √ | √ | √ |
| 0111 | 8 | √ | √ | √ |
| 1000 | 9 | √ | √ | √ |
| Other than above | 10 | √ | √ | √ |

**NOTES**

1. The read access time to the data flash is calculated by the number of wait cycles.

   Read access time to the data flash = {17 + (Number of wait cycles) × 4)} / CPU operating frequency

   However, the time may be changed depending on the combination of instructions before and after the execution.
2. √ indicates the number of wait cycles that can be set.

### 37.8.2.2 PRDNAMEn — Product Name Storage Register (n = 1 to 3)

This register stores the product name. The product part name is stored in 16-byte ASCII code, and PRDNAME1, PRDNAME2, and PRDNAME3 correspond to the fourth to first bytes, eighth to fifth bytes, and twelfth to ninth bytes of the product part name respectively.

**Access:** These registers are read-only registers that can be read in 32-bit units.

**Address:** PRDNAME1: FFCD 00D0$_H$
PRDNAME2: FFCD 00D4$_H$
PRDNAME3: FFCD 00D8$_H$

**Value after reset:** See **Table 37.10** to **Table 37.12**.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PRDNAMEn[31:24]*1 | | | | | | | | PRDNAMEn[23:16]*1 | | | | | | | |
| Value after reset | | | | | | | | | | | | | | | | |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PRDNAMEn[15:8]*1 | | | | | | | | PRDNAMEn[7:0]*1 | | | | | | | |
| Value after reset | | | | | | | | | | | | | | | | |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. n = 1 to 3.

**Table 37.9 PRDNAMEn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | — | Product name fourth byte (PRDNAME1), eighth byte (PRDNAME2) twelfth byte (PRDNAME3) |
| 23 to 16 | — | Product name third byte (PRDNAME1), seventh byte (PRDNAME2) eleventh byte (PRDNAME3) |
| 15 to 8 | — | Product name second byte (PRDNAME1), sixth byte (PRDNAME2) tenth byte (PRDNAME3) |
| 7 to 0 | — | Product name first byte (PRDNAME1), fifth byte (PRDNAME2) ninth byte (PRDNAME3) |

**Table 37.10** to **Table 37.12** list registers related to product information.

**Table 37.10 List of Registers Related to Product Information (PREMIUM)**

| Product Part Name | PRDNAME1 | PRDNAME2 | PRDNAME3 |
|---|---|---|---|
| R7F701620 | 3746 3752 | 3236 3130 | 2020 2030 |
| R7F701621 | 3746 3752 | 3236 3130 | 2020 2031 |
| R7F701622 | 3746 3752 | 3236 3130 | 2020 2032 |
| R7F701623 | 3746 3752 | 3236 3130 | 2020 2033 |
| R7F701580 | 3746 3752 | 3835 3130 | 2020 2030 |
| R7F701581 | 3746 3752 | 3835 3130 | 2020 2031 |
| R7F701582 | 3746 3752 | 3835 3130 | 2020 2032 |
| R7F701583 | 3746 3752 | 3835 3130 | 2020 2033 |
| R7F701586 | 3746 3752 | 3835 3130 | 2020 2036 |
| R7F701587 | 3746 3752 | 3835 3130 | 2020 2037 |
| R7F701597 | 3746 3752 | 3935 3130 | 2020 2037 |

Table 37.11 List of Registers Related to Product Information (ADVANCED)

| Product Part Name | PRDNAME1 | PRDNAME2 | PRDNAME3 |
|---|---|---|---|
| R7F701610 | 3746 3752 | 3136 3130 | 2020 2030 |
| R7F701611 | 3746 3752 | 3136 3130 | 2020 2031 |
| R7F701612 | 3746 3752 | 3136 3130 | 2020 2032 |
| R7F701613 | 3746 3752 | 3136 3130 | 2020 2033 |
| R7F701560 | 3746 3752 | 3635 3130 | 2020 2030 |
| R7F701561 | 3746 3752 | 3635 3130 | 2020 2031 |
| R7F701562 | 3746 3752 | 3635 3130 | 2020 2032 |
| R7F701563 | 3746 3752 | 3635 3130 | 2020 2033 |
| R7F701566 | 3746 3752 | 3635 3130 | 2020 2036 |
| R7F701567 | 3746 3752 | 3635 3130 | 2020 2037 |
| R7F701577 | 3746 3752 | 3735 3130 | 2020 2037 |

Table 37.12 List of Registers Related to Product Information (ECO)

| Product Part Name | PRDNAME1 | PRDNAME2 | PRDNAME3 |
|---|---|---|---|
| R7F701602 | 3746 3752 | 3036 3130 | 2020 2032 |
| R7F701603 | 3746 3752 | 3036 3130 | 2020 2033 |
| R7F701542 | 3746 3752 | 3435 3130 | 2020 2032 |
| R7F701543 | 3746 3752 | 3435 3130 | 2020 2033 |
| R7F701546 | 3746 3752 | 3435 3130 | 2020 2036 |
| R7F701547 | 3746 3752 | 3435 3130 | 2020 2037 |
| R7F701557 | 3746 3752 | 3535 3130 | 2020 2037 |

### 37.8.2.3 CHIPIDnXX — Chip ID Register (n = 1,2, XX = LL,LH,HL,HH)

The RH850/F1K provides the option of a unique chip ID of the device.

**Access:** These registers are read-only registers that can be read in 32-bit units.

**Address:** CHIPID1LL: FFCD 00E0$_H$
CHIPID1LH: FFCD 00E4$_H$
CHIPID1HL: FFCD 00E8$_H$
CHIPID1HH: FFCD 00EC$_H$
CHIPID2LL: FFCD 00F0$_H$
CHIPID2LH: FFCD 00F4$_H$
CHIPID2HL: FFCD 00F8$_H$
CHIPID2HH: FFCD 00FC$_H$

**Value after reset:** Unique value

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CHIPIDnXX[31:24]*1 | | | | | | | | CHIPIDnXX[23:16]*1 | | | | | | | |
| Value after reset | | | | | | | | | | | | | | | | |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CHIPIDnXX[15:8]*1 | | | | | | | | CHIPIDnXX[7:0]*1 | | | | | | | |
| Value after reset | | | | | | | | | | | | | | | | |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. n = 1, 2 XX = LL, LH, HL, HH

## 37.9 Option Bytes

The option bytes of the flash memory are an expansion area and hold data specified by the user for a variety of purposes. Initial settings for peripheral modules and so on as specified by the option bytes become effective on release from the reset state.

### 37.9.1 Option Byte Setting

Be sure to set the option byte area that corresponds to the optional functions listed below, before writing a program to the flash memory.

The optional functions specified by the option bytes are as follows.

- Function of port group JP0
- Activation code method of WDTA1
- Start mode of WDTA1
- Enabling or disabling WDTA1
- Activation code method of WDTA0
- Start mode of WDTA0
- Enabling or disabling WDTA0
- Initial value of the overflow interval time for WDTA1 and WDTA0
- Protocol of CAN FD CRC (Supported only for PREMIUM)
- Enabling or disabling of $\overline{\text{RESETOUT}}$
- Enabling the high voltage monitor
- Enabling the low voltage monitor

### 37.9.2 OPBT0 — Option Byte 0

The settings and bit positions of the option bytes are listed below.

For details on how to set an option byte, see the user's manual of the flash programmer for serial programming as well as the code flash library for self-programming.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | OPJTAG[1:0] | | — | — | WDT 1_3 | — | WDT 1_1 | WDT 1_0 | WDT 0_3 | — | WDT 0_1 | WDT 0_0 | WDT_2 | WDT_1 | WDT_0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | CANFD CRC | RESET OUTEN | — | — | — | CVM_H D_EN | CVM_L D_EN | — | — | — | — |

**Table 37.13 Option Byte 0 Settings (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | When writing, write “1”. |
| 30, 29 | OPJTAG[1:0] | These bits control the function of port group JP0.<br>00: JP0 is used for general purpose/alternative function port.<br>01: JP0 is used for LPD 4-pin mode.<br>10: JP0 is used for LPD 1-pin mode.<br>11: JP0 is used for Nexus I/F. |
| 28, 27 | Reserved | When writing, write “1”. |
| 26 | WDT1_3 | Specifies the activation code method of WDTA1.<br>0: Fixed activation code<br>1: Variable activation code |
| 25 | Reserved | When writing, write “1”. |
| 24 | WDT1_1 | Specifies the start mode of WDTA1.<br>0: Software trigger start mode<br>1: Default start mode |
| 23 | WDT1_0 | Enables or disables WDTA1.<br>0: WDTA1 is disabled<br>1: WDTA1 is enabled |
| 22 | WDT0_3 | Specifies the activation code method of WDTA0.<br>0: Fixed activation code<br>1: Variable activation code |
| 21 | Reserved | When writing, write “1”. |
| 20 | WDT0_1 | Specifies the start mode of WDTA0.<br>0: Software trigger start mode<br>1: Default start mode |
| 19 | WDT0_0 | Enables or disables WDTA0.<br>0: WDTA0 is disabled<br>1: WDTA0 is enabled |
| 18 to 16 | WDT_[2:0] | Control of the overflow interval time for WDTA0 and WDTA1<br>These bits specify the reset value of WDTAnMD.WDTAnOVF[2:0]. |
| 15 to 11 | Reserved | When writing, write “1”. |
| 10 | CANFDCRC*1 | CAN FD CRC protocol<br>0: Original protocol<br>1: New protocol |
| 9 | RESETOUTEN | $\overline{\text{RESETOUT}}$ control<br>0: $\overline{\text{RESETOUT}}$ disable<br>1: $\overline{\text{RESETOUT}}$ enable |
| 8 to 6 | Reserved | When writing, write “1”. |

**Table 37.13 Option Byte 0 Settings (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 5 | CVM_HD_EN | High Voltage Monitor Enable<br>0: Disable high voltage detection<br>1: Enable high voltage detection |
| 4 | CVM_LD_EN | Low Voltage Monitor Enable<br>0: Disable low voltage detection<br>1: Enable low voltage detection |
| 3 to 0 | Reserved | When writing, write “1”. |

Note 1. This bit is supported only for PREMIUM.

### 37.9.3 OPBT1 — Option Byte 1

The settings and bit positions of the option bytes are listed below.

For details on how to set an option byte, see the user's manual of the flash programmer for serial programming as well as the code flash library for self-programming.

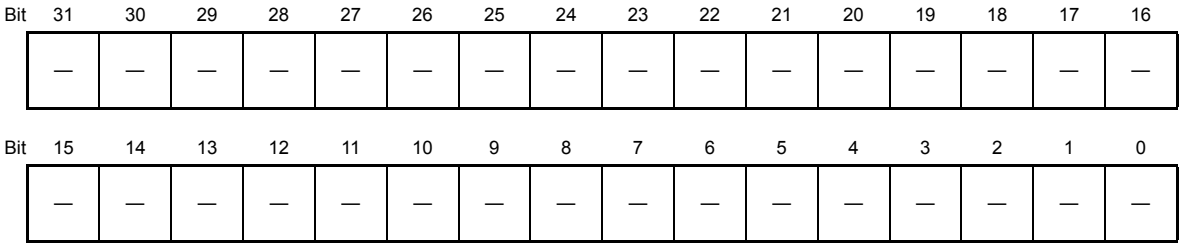



**Table 37.14 Option Byte 1 Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 10 | Reserved | When writing, write “1”. |
| 9 | Reserved | When writing, write “0”. |
| 8 to 0 | Reserved | When writing, write “1”. |

## 37.10 Usage Notes

### (1) Reading areas where programming or erasure was interrupted

When programming or erasure of an area of flash memory is interrupted, the data stored in the area become undefined. To avoid the reading out of undefined data, which might cause a malfunction, take care not to fetch instructions or read data from areas where programming or erasure was interrupted.

### (2) Reading the code flash memory that has been erased but not yet been programmed again

Note that reading from an area of code flash memory that has been erased but not yet been programmed again (i.e. that is in the non-programmed state) can lead to the detection of an ECC error and generation of the corresponding exception. Use blank checking when you need to confirm that an area is in the non-programmed state.

### (3) Prohibition of additional writing

Writing to a given area two or more times is not possible. When overwriting data in an area of flash memory after writing to the area has been completed, erase the area first.

### (4) Resets during programming and erasure

In the case of an external reset during programming and erasure, wait for at least the minimum value of $\overline{\text{RESET}}$ input low level width once the operating voltage is within the range stipulated in the electrical characteristics after assertion of the reset signal before releasing the device from the reset state.

### (5) Allocation of vectors for interrupts and other exceptions during programming and erasure

Generation of an interrupt or other exception during programming or erasure may lead to fetching of the vector from the code flash memory. If this does not satisfy the conditions for using background operation, set the address for vector fetching to an address that is not in the code flash memory.

### (6) Abnormal termination of programming and erasure

Even if programming/erasure ends abnormally due to the assertion of a reset by the $\overline{\text{RESET}}$ pin, the programming/erasure state of the flash memory with undefined data cannot be verified or checked. For the area where programming/erasure ends abnormally, the blank check function cannot judge whether the area is erased successfully or not. Erase the area again to ensure that the corresponding area is completely erased before using.

If programming and erasure of code flash memory are not completed normally, the lock bit for the target area may be enabled (locked). In such cases, erase the block to erase the lock bit while the lock bit is in the disabled state (the area is not locked).

### (7) Items prohibited during programming and erasure

Do not perform the following operations during programming and erasure.

- Set the operating voltage from the power supply outside the allowed range.
- Change the frequency of the peripheral clock.

### (8) Erasure of all flash memory before clearing the configuration

Erase the user area of code flash and all of the data area of data flash before clearing the configuration.

### (9) Restriction on serial programming when HS IntOSC is used

- 1-wire/2-wire UART mode cannot be used.
- The E1 emulator cannot be used.

# Section 38 RAM

This section describes the local RAM mounted on RH850/F1K.

## 38.1 Features

- RH850/F1K includes the following RAMs:
  - Local RAM
    The local RAM is accessible at high speed. Values in the local RAM are not retained in DeepSTOP mode.
  - Retention RAM
    The retention RAM is a part of the local RAM and is also accessible at high speed. Values in the retention RAM are retained in DeepSTOP mode.
    In addition except for the write access, even if the power-supply voltage (REGVCC) falls below the POC voltage, data in the retention RAM are retained as long as the voltage does not fall below the RAM retention voltage ($V_{VLVI}$).

Access time for each RAM is shown in the table below.

**Table 38.1 RAM Access Time**

| Type of Access | RAM | 1st Access (CPUCLK) | Continuous Access (CPUCLK) |
|---|---|---|---|
| Instruction fetch | Local RAM | 5 | 5 |
| | Retention RAM | 5 | 5 |
| Read access | Local RAM | 1 | 1 |
| | Retention RAM | 1 | 1 |
| Write access | Local RAM | 1 | 1 |
| | Retention RAM | 1 | 1 |

**NOTE**

There is possibility that number of access clock of above table is changed depending on the combination before and after instructions.

When RAM access is misaligned, these number is increased.

- Error detection/correction function (ECC) in the local RAM and retention RAM
  The ECC function is included, which can detect 2-bit errors and detect/correct 1-bit errors. For details, see **Section 33, Functional Safety**.

## 38.2 Memory Configuration

**Figure 38.1** shows the memory map of the local RAM and the retention RAM.

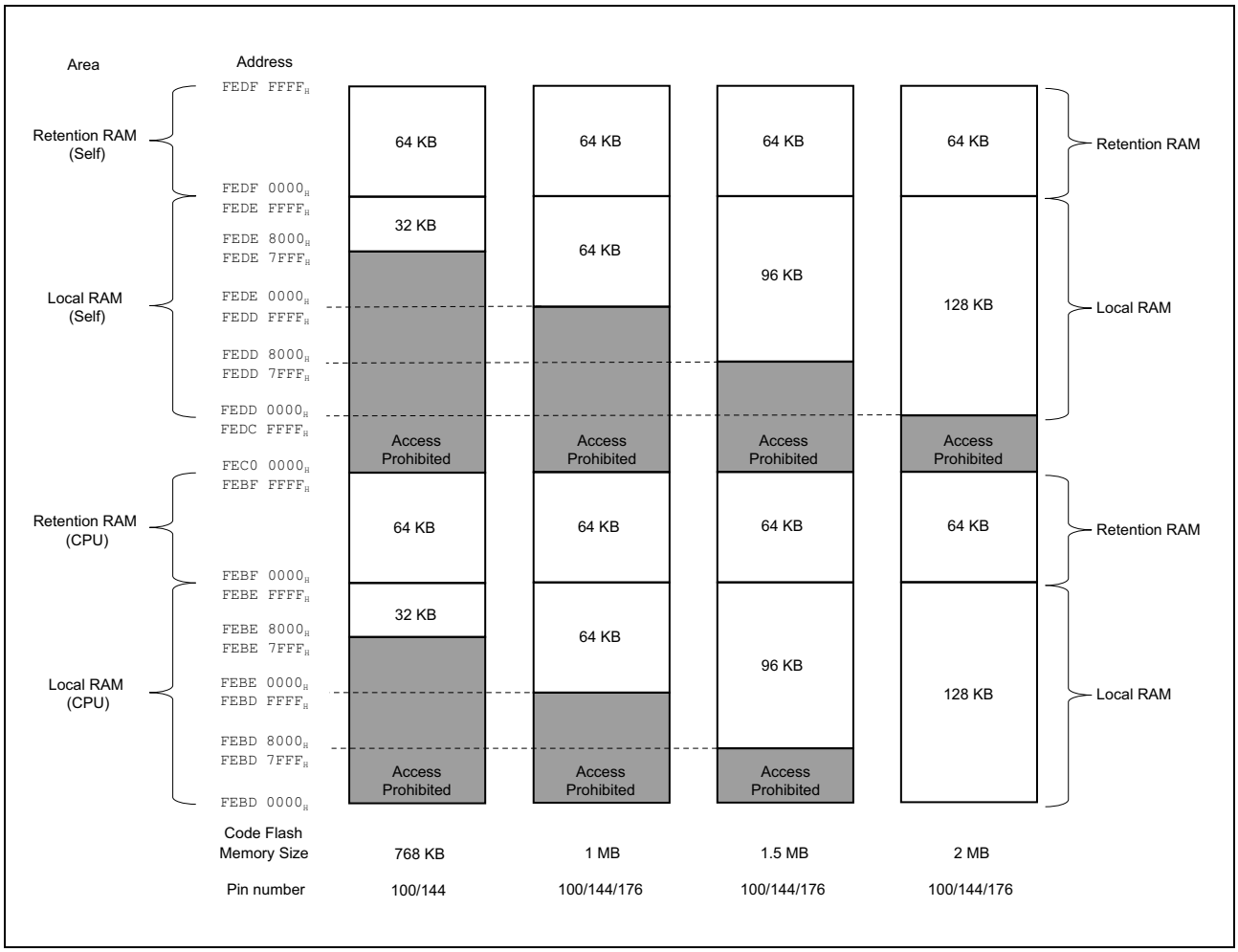



**Figure 38.1 Memory Map of the Local RAM and the Retention RAM**

## 38.3 Usage Notes

Before accessing the local or retention RAM with ECC error detection and correction enabled, initialize the RAM by setting the access size to the largest bit length.

Accessing the RAM before initializing it may lead to the detection of ECC errors. ECC errors may be detected if initialization is not handled in the maximum unit of access, for example, when 32-bit word RAM is accessed in 8- or 16-bit units.

# Section 39 Boundary Scan

This section contains a generic description of boundary scan.

The RH850/F1K has a JTAG interface and provides a boundary scan function.

## 39.1 Overview

Boundary scan is a test method defined in the IEEE standard 1149.1 that is used to test the connection between the devices mounted on the printed-circuit board. The boundary scan of the RH850/F1K conforms to IEEE Std 1149.1-2001.

## 39.2 Features

- Five control signals (DCUTCK, DCUTDI, DCUTDO, DCUTMS, and $\overline{\text{DCUTRST}}$)
- TAP controller
- Instruction register
- Bypass register
- Boundary scan register

The JTAG interface has four instructions.

- BYPASS
  Test mode conforming to the IEEE 1149.1
- EXTEST
  Test mode conforming to the IEEE 1149.1
- SAMPLE/PRELOAD
  Test mode conforming to the IEEE 1149.1
- IDCODE
  Test mode conforming to the IEEE 1149.1

**Figure 39.1** shows a block diagram of the JTAG interface.

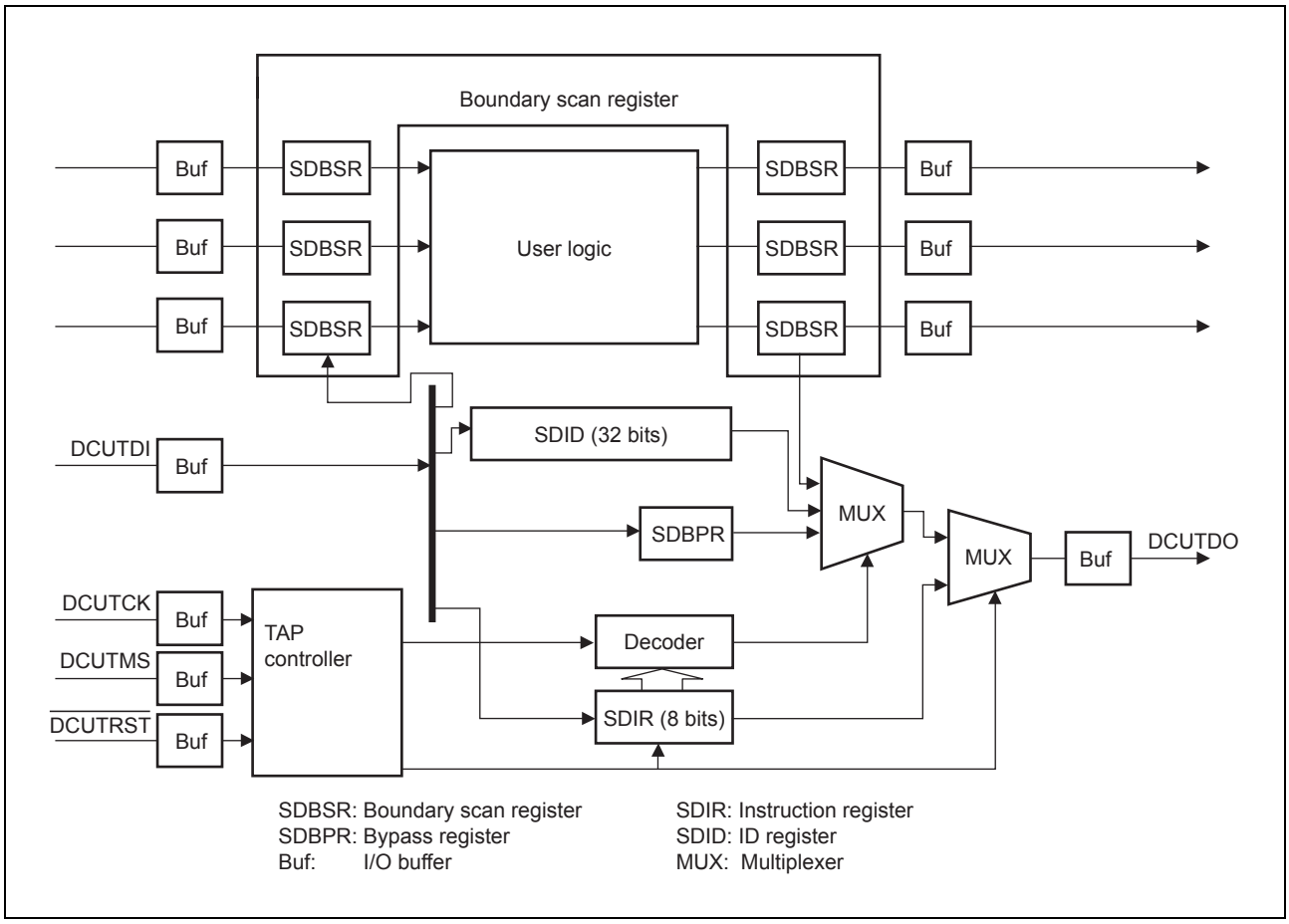



**Figure 39.1 Block Diagram of JTAG Interface**

## 39.3 External Input/Output Pins

There are five JTAG control signals: DCUTCK, DCUTDI, DCUTMS, DCUTDO, and $\overline{\text{DCUTRST}}$.

**Table 39.1** shows the pin configuration.

**Table 39.1 Pin Configuration**

| Pin Name | Description |
|---|---|
| DCUTCK | Serial data input/output clock pin<br>Data is input to DCUTDI and is output from DCUTDO in synchronization with this clock signal. |
| DCUTMS | Mode select input pin<br>Changing the level of this signal in synchronization with DCUTCK changes the state of the TAP controller. For the protocol, see **Figure 39.2, TAP Controller State Transition Diagram**. |
| $\overline{\text{DCUTRST}}$ | Reset input pin<br>A low-level input of this signal resets the JTAG interface. This signal is accepted asynchronously with DCUTCK. |
| DCUTDI | Serial data input pin<br>Data is input in synchronization with DCUTCK and sent to the JTAG interface. |
| DCUTDO | Serial data output pin<br>Data to be read from the JTAG interface is output in synchronization with DCUTCK. |

## 39.4 Register Descriptions

The JTAG interface has the following registers. None of the registers can be accessed by the CPU.

- SDIR: Instruction register
- SDID: ID register
- SDBPR: Bypass register
- SDBSR: Boundary scan register

**Table 39.2 Register Configuration**

| Register Name | Symbol | Access Size | Value After Reset*1 |
|---|---|---|---|
| Instruction register | SDIR | 8 | $55_H$ |
| ID register | SDID | 32 | See **Table 39.5, ID Register Codes** |
| Bypass register | SDBPR | 1 | Undefined |
| Boundary scan register | SDBSR | — | Undefined |

Note 1. Registers are initialized when $\overline{\text{DCUTRST}}$ pin is 0 or when TAP is in the Test-Logic-Reset state.

Instructions can be serially transferred from the serial data input pin (DCUTDI) and input to the instruction register (SDIR). The bypass register (SDBPR) is a 1-bit register, to which DCUTDI and DCUTDO are connected in BYPASS mode. The boundary scan register (SDBSR) is connected to DCUTDI and DCUTDO in SAMPLE/PRELOAD mode and EXTEST mode. The ID register (SDID) is a 32-bit register, from which the fixed code is output via DCUTDO in IDCODE mode.

**Table 39.3** shows the serial transfer types possible with the JTAG interface registers.

**Table 39.3 Serial Transfer Types**

| Register | Serial Input | Serial Output |
|---|---|---|
| SDIR | Possible | Impossible*1 |
| SDBPR | Possible | Possible |
| SDBSR | Possible | Possible |
| SDID | Impossible | Possible |

Note 1. A fixed value is read out.

### 39.4.1 Instruction Register (SDIR)

The instruction register (SDIR) is an 8-bit register that holds a boundary scan instruction. SDIR is initialized by a low-level input of $\overline{\text{DCUTRST}}$ or in the TAP Test-Logic-Reset state. Operation is not guaranteed if a reserved instruction is set in this register.

**Table 39.4 Boundary Scan Instructions**

| Instruction Code | | | | | | | | Description |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | JTAG EXTEST |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | JTAG SAMPLE/PRELOAD |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | JTAG IDCODE (initial value) |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | JTAG BYPASS |
| Other than above | | | | | | | | Reserved |

### 39.4.2 ID Register (SDID)

The ID register (SDID) is a 32-bit register with a device specific ID.

SDID can be read from the JTAG interface when the IDCODE instruction is set, but cannot be accessed from the CPU.

For the read values, see **Table 39.2, Register Configuration** and **Table 39.5, ID Register Codes**.

**Table 39.5 ID Register Codes**

| PKG | ID Register Codes | | | |
|---|---|---|---|---|
| | 31 to 28 | 27 to 12 | 11 to 1 | 0 |
| 176 pins | 0001 | 1000 0011 0000 1000 | 0100 0100 011 | 1 |
| 144 pins | 0001 | 1000 0011 0001 0010 | 0100 0100 011 | 1 |
| 100 pins | 0001 | 1000 0011 0001 0011 | 0100 0100 011 | 1 |

### 39.4.3 Bypass Register (SDBPR)

The bypass register (SDBPR) is a 1-bit register. When SDIR is set to BYPASS mode, SDBPR is connected to the position between DCUTDI and DCUTDO. The value after reset is undefined. SDBPR is not initialized by a power-on reset or by a low-level input of $\overline{\text{DCUTRST}}$.

### 39.4.4 Boundary Scan Register (SDBSR)

The boundary scan register (SDBSR) is a shift register for controlling the external I/O pins. When SDIR is set to SAMPLE/PRELOAD or EXTEST mode, SDBSR is connected to the position between DCUTDI and DCUTDO. The value after reset is undefined. SDBSR is not initialized by a power-on reset or a low-level input of $\overline{\text{DCUTRST}}$.

## 39.5 Operation

### 39.5.1 TAP Controller

**Table 39.2** shows the state transition of the TAP controller. The transition condition is the DCUTMS value at the rising edge of DCUTCK.

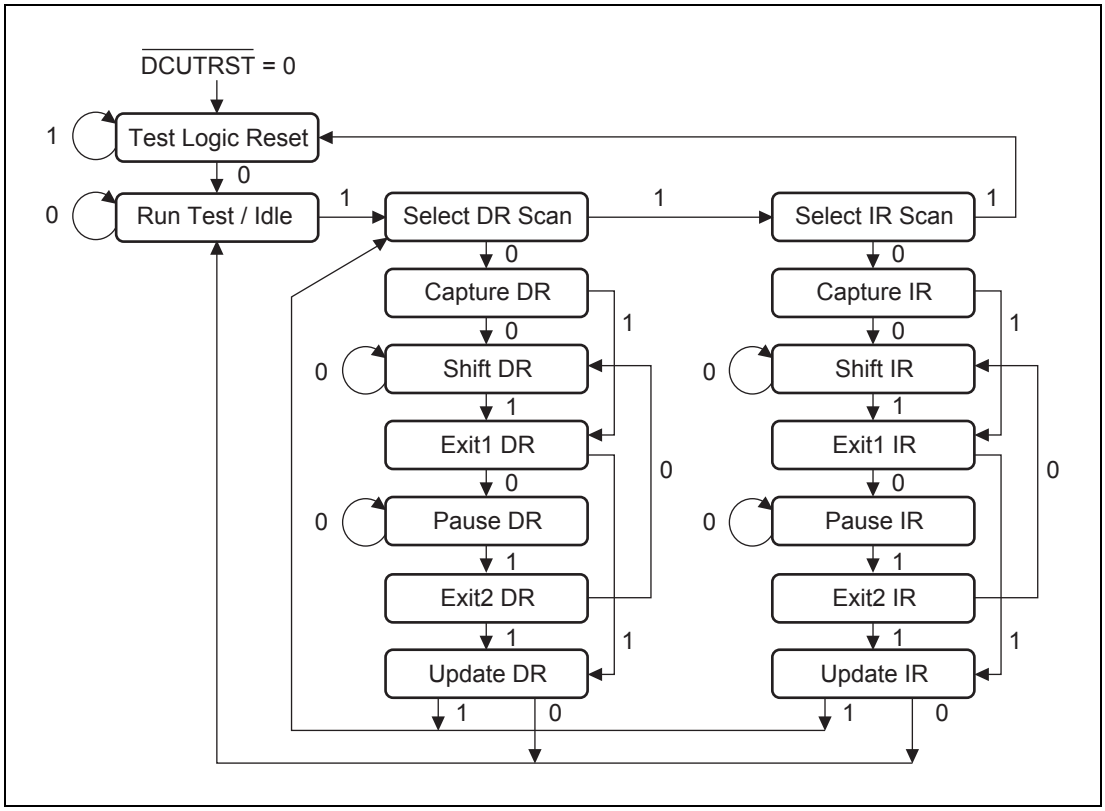



**Figure 39.2 TAP Controller State Transition Diagram**

**NOTE**

The DCUTDI value is sampled at the rising edge of DCUTCK and is shifted at the falling edge. DCUTDO is in the high-impedance state in the states other than Shift-DR and Shift-IR. A low-level input of $\overline{\text{DCUTRST}}$ causes transition to Test-Logic-Reset state asynchronously with DCUTCK.

## 39.5.2 Supported Instructions

### 39.5.2.1 BYPASS

The BYPASS instruction is a required standard instruction to operate the bypass register. This instruction shortens the shift path to achieve high-speed serial data transfer of other devices on the printed-circuit board. During execution of this instruction, the test circuit has no effect on the system circuit.

### 39.5.2.2 SAMPLE/PRELOAD

The SAMPLE/PRELOAD instruction is used to input the value to the boundary scan register from the internal circuits of this device; to output the value from the scan path; and to load data onto the scan path. During execution of this instruction, the level of the input pin of this device is sent to the internal circuits as is, and the value of the internal circuits is output to the outside via the output pin as is. Executing this instruction has no effect on the system circuit of this device.

The SAMPLE operation allows taking in the snapshots of the value to be transferred to the internal circuits from the input pin or the value to be transferred to the output pin from the internal circuits to the boundary scan register and allows reading the snapshots from the scan path. Snapshots can be taken in without preventing the normal operation of this device.

The PRELOAD operation allows setting the initial value to the parallel output latch of the boundary scan register from the scan path prior to the EXTEST instruction. If the EXTEST instruction is executed without PRELOAD operation, an undefined value is output from the output pin until the first scan sequence is completed (transfer to the output latch) because the parallel output latch value is always output to the output pin by the EXTEST instruction.

### 39.5.2.3 EXTEST

The EXTEST instruction is used to test the external circuits when this device is mounted on the printed-circuit board. When this instruction is executed, the output pin is used to output the test data (previously set by the SAMPLE/PRELOAD instruction) from the boundary scan register to the printed-circuit board; whereas the input pin is used to take in the test result from the printed-circuit board to the boundary scan register. When the EXTEST instruction is executed N times for testing, the test data for the Nth execution is scanned in at the (N - 1)th scan-out.

If the data is loaded onto the boundary scan register of the output pin in the Capture-DR state of this instruction, it is not used for testing the external circuits (replaced through shift operation).

### 39.5.2.4 IDCODE

The IDCODE instruction sets the JTAG interface pins to IDCODE mode, which is defined by the JTAG standard. When the JTAG interface is initialized (by a low-level input of $\overline{\text{DCUTRST}}$ or placing TAP in the Test-Logic-Reset state), IDCODE mode is set.

### 39.5.3 Pins Subject to Boundary Scan

All pins, excluding pins such as external clock input pins or power supply pins, are subject to boundary scan.

The pins which are not subjected to boundary scan are listed in **Table 39.6**.

**Table 39.6 Pins not Subject to Boundary Scan**

| Type | Pins |
|---|---|
| JTAG interface | DCUTCK, DCUTDI, DCUTDO, DCUTMS, $\overline{\text{DCUTRST}}$ |
| Power supply pins | REGVCC, AWOVCL, AWOVSS, ISOVCL, ISOVSS<br>EVCC, EVSS |
| Power supply pins (A/D converter) | A0VREF, A1VREF*1, A0VSS, A1VSS*1 |
| Clock | X1, X2, XT1*1, IP0_0/XT2*1 |
| Mode setting | P10_8/FLMD1, P10_1/MODE0, P10_2/MODE1 |

Note 1. Only available for 176 pin and 144 pin devices.

The following signals are only sampled in boundary scan mode.

**Table 39.7 Pins Subject to Boundary Scan (Sampling Only)**

| Function | Pin Name |
|---|---|
| Reset | $\overline{\text{RESET}}$ |
| Mode setting | FLMD0 |

The following pins are shared by the analog buffer. Accordingly, boundary scan only applies to general I/O pins.

**Table 39.8 Pins Subject to Boundary Scan (Only General I/O Pins)**

| Function | Pin Name |
|---|---|
| ADCA0 input | P8_0-12, P9_0-6, AP0_0-15 |
| ADCA1 input | P18_0-7, AP1_0-15 |

**NOTE**

In boundary scan mode, the level of the following pins must be fixed:
P10_1 = Low, P10_2 = High, and P10_8 = High

## 39.6 Usage Notes

1. Once an instruction is set, it is not modified until another instruction is issued. To issue the same instruction twice in a row, insert an instruction that has no effect on chip operation (such as BYPASS) between the instructions.
2. To start the system in boundary scan mode, de-assert $\overline{\text{DCUTRST}}$ while $\overline{\text{RESET}}$ is high. Also be sure to set DCUTMS to high before de-asserting $\overline{\text{DCUTRST}}$ and ensure that DCUTMS remains high for 600 ns + five DCUTCK clock cycles after de-asserting $\overline{\text{DCUTRST}}$.
3. For the maximum clock frequency that can be input to DCUTCK, see the Electrical Characteristics section in the Data Sheet document.
4. If serial transfer is performed exceeding the number of bits of the register connected between DCUTDI and DCUTDO, the data that is input from DCUTDI is output from DCUTDO as is.
5. If the serial transfer sequence is corrupted, be sure to assert $\overline{\text{DCUTRST}}$. In this case, transfer starts again from the beginning regardless of the point of transfer corruption.
6. Data is output via DCUTDO at the falling edge of DCUTCK.
7. To facilitate debugging, route $\overline{\text{DCUTRST}}$ on the board in such a way that patterns can be easily cut.

# Section 40 Electrical Characteristics

## 40.1 Overview

The electrical spec of this device is guaranteed by the following operational condition. But, this condition is different depends on each characteristics, so refer to each chapter for more detail.

### 40.1.1 Pin Groups

#### 40.1.1.1 176 pin

| Symbol | Pin Group Supplied by | Related Pins/Ports |
|---|---|---|
| PgR | REGVCC, AWOVSS | X1, X2, XT1, XT2/IP0_0 |
| PgE | EVCC, EVSS | Related ports: JP0, P0, P1, P2, P8, P9, P10, P11, P12, P18, P20<br>Related pins: $\overline{\text{RESET}}$, FLMD0 |
| PgA0 | A0VREF, A0VSS | Related port: AP0 |
| PgA1 | A1VREF, A1VSS | Related port: AP1 |

#### 40.1.1.2 144 pin

| Symbol | Pin Group Supplied by | Related Pins/Ports |
|---|---|---|
| PgR | REGVCC, AWOVSS | X1, X2, XT1, XT2/IP0_0 |
| PgE | EVCC, EVSS | Related ports: JP0, P0, P1, P8, P9, P10, P11, P12, P18, P20<br>Related pins: $\overline{\text{RESET}}$, FLMD0 |
| PgA0 | A0VREF, A0VSS | Related port: AP0 |
| PgA1 | A1VREF, A1VSS | Related port: AP1 |

#### 40.1.1.3 100 pin

| Symbol | Pin Group Supplied by | Related Pins/Ports |
|---|---|---|
| PgR | REGVCC, AWOVSS | X1, X2 |
| PgE | EVCC, EVSS | Related ports: JP0, P0, P8, P9, P10, P11<br>Related pins: $\overline{\text{RESET}}$, FLMD0 |
| PgA0 | A0VREF, A0VSS | Related port: AP0 |

### 40.1.2 General Measurement Conditions

#### 40.1.2.1 Common Conditions

- Power supply
  - REGVCC = EVCC = VPOC*1 to 5.5 V
  - A0VREF = 3.0 V to 5.5 V
  - A1VREF = 3.0 V to 5.5 V
  - AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V

- Capacitance of the internal regulator
  - CAWOVCL: 0.1 µF ± 30%
  - CISOVCL: 0.1 µF ± 30%
- Operating temperature
  - Tj = –40 to +130°C @R7F7015xx3AFP*
    @R7F7016yy3AFP*
  - Tj = –40 to +150°C @R7F7015xx4AFP*
    @R7F7016yy4AFP*

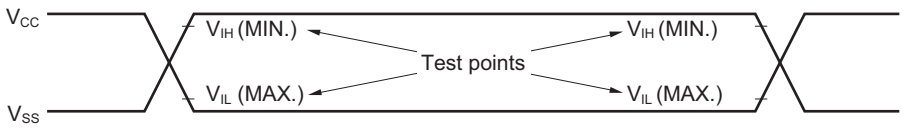
  xx = 42, 43, 46, 47, 57, 60, 61, 62, 63, 66, 67, 77, 80, 81, 82, 83, 86, 87, 97
  yy = 02, 03, 10, 11, 12, 13, 20, 21, 22, 23

  Regarding operation temperature of each product, see **Section 1.3, RH850/F1K Product Lineup**.
- Load conditions
  - CL = 30 pF

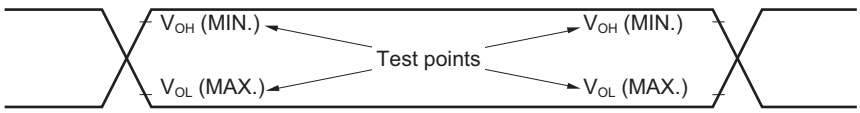
**Note 1.** "VPOC" means POC (power on clear) detection voltage. For more detail, see **Section 40.8.2, Voltage Detector (POC, LVI, VLVI, CVM) Characteristics**.

### 40.1.2.2 AC Characteristic Measurement Condition

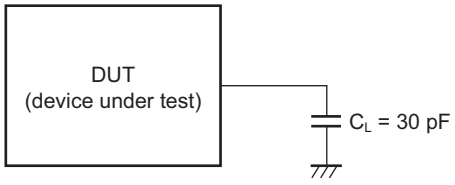
#### (1) AC test input measurement points




#### (2) AC test output measurement points




#### (3) Load conditions



**CAUTION**

**If the load capacitance exceeds 30pF due to the circuit configuration, it is recommended to insert a buffer in order to reduce capacitance to less than 30pF.**

## 40.2 Absolute Maximum Ratings

**CAUTIONS**

1. **Do not directly connect outputs (or input/outputs) to each other, power supply and ground.**
2. **Even momentarily exceeding the absolute maximum rating for just one item creates a threat of failure in the reliability of the products. That is, the absolute maximum ratings are the levels that raise a threat of physical damage to the products. Be sure to use the products only under conditions that do not exceed the ratings. The quality and normal operation of the product are guaranteed under the standards and conditions given as DC and AC characteristics.**
3. **When designing an external circuit ensure that the connections don't conflict with the port state of this device.**

### 40.2.1 Supply Voltages

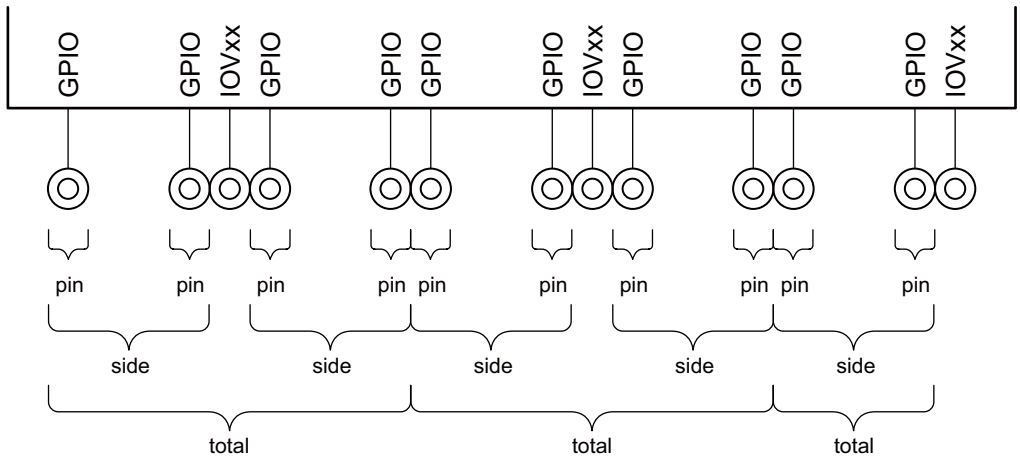
| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| System supply voltage | REGVCC | | –0.5 | | 6.5 | V |
| | AWOVSS | | –0.5 | | 0.5 | V |
| | ISOVSS | | –0.5 | | 0.5 | V |
| Port supply voltage | EVCC | | –0.5 | | 6.5 | V |
| | EVSS | | –0.5 | | 0.5 | V |
| A/D-converter supply voltage | A0VREF | | –0.5 | | 6.5 | V |
| | A1VREF | | –0.5 | | 6.5 | V |
| | A0VSS | | –0.5 | | 0.5 | V |
| | A1VSS | | –0.5 | | 0.5 | V |

### 40.2.2 Port Voltages

| Item | Pin Group*1 | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Input voltage | PgR | VI | | –0.5 | | REGVCC + 0.5 (Do not exceed 6.5 V) | V |
| | PgE | | | –0.5 | | EVCC + 0.5 (Do not exceed 6.5 V) | V |
| | PgA0 | | | –0.5 | | A0VREF + 0.5 (Do not exceed 6.5 V) | V |
| | PgA1 | | | –0.5 | | A1VREF + 0.5 (Do not exceed 6.5 V) | V |

Note 1. The characteristics of the alternative-function pins are the same as those of the port pins unless otherwise specified.

### 40.2.3 Port Current




**Definition of the condition:**

- **Per pin:** Output current of one GPIO
- **Per side:** Total output current of all GPIO pins on one side of one IOVxx
- **Total:** Total output current of both sides of one IOVxx

**Note:**

- **GPIO:** General-purpose I/O pin (JP0, P0, P1, P2, P8, P9, P10, P11, P12, P18, P20, AP0, AP1)
- **IOVxx:** Power supply pin for I/O pins (EVCC/EVSS, A0VREF/A0VSS, A1VREF/A1VSS)

### 40.2.3.1 176 pin

| Item | Symbol | Pin Group | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| High-level output current | IOH | PgE | Per pin | | | –10 | mA |
| | | | Per side1 (Total of P9_0 to P9_6, P20_0 to P20_5) | | | –48 | mA |
| | | | Per side2 (Total of P10_6 to P10_9, P18_0 to P18_7) | | | –48 | mA |
| | | | Per side3 (Total of P10_10 to P10_14, P11_1 to P11_7, P11_15, P12_0 to P12_2) | | | –48 | mA |
| | | | Per side4 (Total of P10_0 to P10_5) | | | –48 | mA |
| | | | Per side5 (Total of P0_0 to P0_3, P10_15, P11_0, P11_8 to P11_14, P12_3 to P12_5) | | | –48 | mA |
| | | | Per side6 (Total of JP0_3 to JP0_5, P0_4 to P0_6, P0_11 to P0_14, P1_0 to P1_3, P1_12, P1_13, P2_6, P8_2, P8_10 to P8_12) | | | –48 | mA |
| | | | Per side7 (Total of JP0_0 to JP0_2, P1_8 to P1_11, P2_0, P2_1) | | | –48 | mA |
| | | | Per side8 (Total of JP0_6, P0_7 to P0_10, P1_4 to P1_7, P1_14, P1_15, P2_2 to P2_5, P8_0, P8_1, P8_3 to P8_9) | | | –48 | mA |
| | | | Total (EVCC: side1 to side4.) | | | –60 | mA |
| | | | Total (EVCC: side4 to side8.) | | | –60 | mA |
| | | PgA0 | Per pin | | | –10 | mA |
| | | | Total (A0VREF) | | | –48 | mA |
| | | PgA1 | Per pin | | | –10 | mA |
| | | | Total (A1VREF) | | | –48 | mA |
| Low-level output current | IOL | PgE | Per pin | | | 10 | mA |
| | | | Per side1 (Total of P9_0 to P9_6, P20_0 to P20_5) | | | 48 | mA |
| | | | Per side2 (Total of P18_0 to P18_7) | | | 48 | mA |
| | | | Per side3 (Total of P10_6 to P10_14, P11_1, P11_2) | | | 48 | mA |
| | | | Per side4 (Total of P11_3 to P11_7, P11_15, P12_0 to P12_2) | | | 48 | mA |
| | | | Per side5 (Total of P10_0 to P10_5) | | | 48 | mA |
| | | | Per side6 (Total of P10_15, P11_0, P11_8 to P11_14, P12_3 to p12_5) | | | 48 | mA |
| | | | Per side7 (Total of P0_0 to P0_6, P0_11 to P0_14, P1_0 to P1_3, P1_12, P1_13, P2_6) | | | 48 | mA |
| | | | Per side8 (Total of JP0_0 to JP0_5, P1_8 to P1_11, P2_0, P2_1, P8_2, P8_10 to P8_12) | | | 48 | mA |
| | | | Per side9 (Total of JP0_6, P0_7 to P0_10, P2_2, P2_3) | | | 48 | mA |
| | | | Per side10 (Total of P1_4 to P1_7, P1_14, P1_15, P2_4, P2_5, P8_0, P8_1, P8_3 to P8_9) | | | 48 | mA |
| | | | Total (EVCC: side1 to side5.) | | | 60 | mA |
| | | | Total (EVCC: side5 to side10.) | | | 60 | mA |
| | | PgA0 | Per pin | | | 10 | mA |
| | | | Total (A0VSS) | | | 48 | mA |
| | | PgA1 | Per pin | | | 10 | mA |
| | | | Total (A1VSS) | | | 48 | mA |

### 40.2.3.2 144 pin

| Item | Symbol | Pin Group | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| High-level output current | IOH | PgE | Per pin | | | –10 | mA |
| | | | Per side1 (Total of P9_0 to P9_6, P20_4, P20_5) | | | –48 | mA |
| | | | Per side2 (Total of P10_6 to P10_9, P18_0 to P18_3) | | | –48 | mA |
| | | | Per side3 (Total of P10_10 to P10_14, P11_1 to P11_7, P11_15, P12_0 to P12_2) | | | –48 | mA |
| | | | Per side4 (Total of P0_0 to P0_3, P10_0 to P10_5, P10_15, P11_0, P11_8 to P11_14) | | | –48 | mA |
| | | | Per side5 (Total of JP0_3 to JP0_5, P0_4 to P0_6, P0_11 to P0_14, P1_0 to P1_3, P8_2, P8_10 to P8_12) | | | –48 | mA |
| | | | Per side6 (Total of JP0_0 to JP0_2, P1_8 to P1_11) | | | –48 | mA |
| | | | Per side7 (Total of JP0_6, P0_7 to P0_10, P1_4 to P1_7, P8_0, P8_1, P8_3 to P8_9) | | | –48 | mA |
| | | | Total (EVCC: side1 to side4.) | | | –60 | mA |
| | | | Total (EVCC: side4 to side7.) | | | –60 | mA |
| | | PgA0 | Per pin | | | –10 | mA |
| | | | Total (A0VREF) | | | –48 | mA |
| | | PgA1 | Per pin | | | –10 | mA |
| | | | Total (A1VREF) | | | –48 | mA |
| Low-level output current | IOL | PgE | Per pin | | | 10 | mA |
| | | | Per side1 (Total of P9_0 to P9_6, P20_4, P20_5) | | | 48 | mA |
| | | | Per side2 (Total of P18_0 to P18_3) | | | 40 | mA |
| | | | Per side3 (Total of P10_6 to P10_14, P11_1, P11_2) | | | 48 | mA |
| | | | Per side4 (Total of P11_3 to P11_7, P11_15, P12_0 to P12_2) | | | 48 | mA |
| | | | Per side5 (Total of P0_0 to P0_6, P0_11 to P0_14, P1_0 to P1_3, P10_0 to P10_5, P10_15, P11_0, P11_8 to P11_14) | | | 48 | mA |
| | | | Per side6 (Total of JP0_0 to JP0_5, P1_8 to P1_11, P8_2, P8_10 to P8_12) | | | 48 | mA |
| | | | Per side7 (Total of JP0_6, P0_7 to P0_10) | | | 48 | mA |
| | | | Per side8 (Total of P1_4 to P1_7, P8_0, P8_1, P8_3 to P8_9) | | | 48 | mA |
| | | | Total (EVCC: side1 to side5.) | | | 60 | mA |
| | | | Total (EVCC: side5 to side8.) | | | 60 | mA |
| | | PgA0 | Per pin | | | 10 | mA |
| | | | Total (A0VSS) | | | 48 | mA |
| | | PgA1 | Per pin | | | 10 | mA |
| | | | Total (A1VSS) | | | 48 | mA |

#### 40.2.3.3 100 pin

| Item | Symbol | Pin Group | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| High-level output current | IOH | PgE | Per pin | | | –10 | mA |
| | | | Per side (Total of P9_0 to P9_6) | | | –48 | mA |
| | | | Per side (Total of P0_0 to P0_3, P10_3 to P10_5, P10_15, P11_0) | | | –48 | mA |
| | | | Per side (Total of JP0_3 to JP0_5, P0_4 to P0_6, P0_11 to P0_14, P8_2, P8_10 to P8_12) | | | –48 | mA |
| | | | Per side (Total of JP0_0 to JP0_2) | | | –30 | mA |
| | | | Per side (Total of P0_7 to P0_10, P8_0, P8_1, P8_3 to P8_9) | | | –48 | mA |
| | | | Per side (Total of P10_6 to P10_9) | | | –40 | mA |
| | | | Per side(Total of P10_10 to P10_14, P11_1 to P11_7) | | | –48 | mA |
| | | | Per side (Total of P10_0 to P10_2) | | | –30 | mA |
| | | | Total (EVCC) | | | –60 | mA |
| | | PgA0 | Per pin | | | –10 | mA |
| | | | Total (A0VREF) | | | –48 | mA |
| Low-level output current | IOL | PgE | Per pin | | | 10 | mA |
| | | | Per side (Total of P9_0 to P9_6) | | | 48 | mA |
| | | | Per side (Total of P0_0 to P0_6, P0_11 to P0_14, P10_3 to P10_5, P10_15, P11_0) | | | 48 | mA |
| | | | Per side (Total of JP0_0 to JP0_5, P8_2, P8_10 to P8_12) | | | 48 | mA |
| | | | Per side (Total of P0_7 to P0_10) | | | 40 | mA |
| | | | Per side (Total of P8_0, P8_1, P8_3 to P8_9) | | | 48 | mA |
| | | | Per side(Total of P10_6 to P10_14, P11_1, P11_2) | | | 48 | mA |
| | | | Per side (Total of P11_3 to P11_7) | | | 48 | mA |
| | | | Per side (Total of P10_0 to P10_2) | | | 30 | mA |
| | | | Total (EVCC) | | | 60 | mA |
| | | PgA0 | Per pin | | | 10 | mA |
| | | | Total (A0VSS) | | | 48 | mA |

### 40.2.4 Temperature Condition

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Storage temperature | Tstg | | –55 | | 170 | °C |
| Junction temperature | Tj | R7F7015xx3AFP*<br>R7F7016yy3AFP* | –40 | | 130 | °C |
| | | R7F7015xx4AFP*<br>R7F7016yy4AFP* | –40 | | 150 | °C |

xx = 42, 43, 46, 47, 57, 60, 61, 62, 63, 66, 67, 77, 80, 81, 82, 83, 86, 87, 97
yy = 02, 03, 10, 11, 12, 13, 20, 21, 22, 23

Regarding operation temperature of each product, see **Section 1.3, RH850/F1K Product Lineup**.

## 40.3 Capacitance

**Condition:** REGVCC = EVCC = A0VREF = A1VREF = AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V, Ta = 25°C

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Input capacitance | CI*1 | f = 1 MHz<br>0 V for non measurement pins | | | 10 | pF |
| Input/output capacitance | CIO*2 | | | | 10 | pF |

Note 1. CI: Capacitance between the input pin and ground

Note 2. CIO: Capacitance between the input/output pin and ground

## 40.4 Operational Condition

### ECO

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| CPU clock frequency | $f_{CPUCLK}$ | | | | 80 | MHz |
| | $f_{CPUCLK2}$ | for OSTMn | | | 40 | MHz |
| Peripheral clock (clock domain) frequency*1 | $f_{CKSCLK_AWDTA}$ | for WDTA0 | | | 240*2 | kHz |
| | $f_{CKSCLK_ATAUJ}$ | for TAUJ0 | | | 40 | MHz |
| | $f_{CKSCLK_ARTCA}$ | for RTCA0 | | | 4 | MHz |
| | $f_{CKSCLK_AADCA}$ | for ADCA0 | | | 40 | MHz |
| | $f_{CKSCLK_AFOUT}$ | for FOUT | | | 24 | MHz |
| | $f_{CKSCLK_IPERI1}$ | for TAUD0 | | | 80 | MHz |
| | | for TAUJ1 | | | | |
| | | for ENCA0 | | | | |
| | | for TAPA0 | | | | |
| | | for PIC0 | | | | |
| | $f_{CKSCLK_IPERI2}$ | for TAUBn | | | 40 | MHz |
| | | for RS-CANn (clkc) | | | | |
| | | for PWBAn | | | | |
| | | for PWGAn | | | | |
| | | for PWSAn | | | | |
| | $f_{CKSCLK_ILIN}$ | for RLIN2m | | | 40 | MHz |
| | | for RLIN3n | | | | |
| | $f_{CKSCLK_IADCA}$ | for ADCA1 | | | 40 | MHz |
| | $f_{CKSCLK_ICANOSC}$ | for RS-CANn (clk_xincan) | | | 24 | MHz |
| | $f_{CKSCLK_ICSI}$ | for CSIGn | | | 80 | MHz |
| | | for CSIHn | | | | |
| | $f_{LS\ IntOSC}$ | for WDTA1 | | | 240*2 | kHz |
| | $f_{CKSCLK_IIIC}$ | for RIICn | | | 40 | MHz |
| | $f_{EMCLK}$ | for LPSn | | | 8 | MHz |
| Power supply | REGVCC | REGVCC = EVCC | VPOC*3 | | 5.5 | V |
| | EVCC | | | | | |
| | A0VREF | | 3.0 | | 5.5 | V |
| | A1VREF | | | | | |
| Normal operation voltage | AWOVCL | | 1.1 | 1.25 | 1.35 | V |
| | ISOVCL | | | | | |
| Limited operation voltage*4 | AWOVCL | | 1.35 | | 1.43 | V |
| | ISOVCL | | | | | |

Note 1. For clock specification of peripherals, see **Section 12, Clock Controller**.

Note 2. This frequency depends on the internal oscillator (LS IntOSC).

Note 3. “VPOC” means POC (power on clear) detection voltage (TYP. 2.85 V). For detail, see **Section 40.8.2, Voltage Detector (POC, LVI, VLVI, CVM) Characteristics**.
In addition, the guaranteed operation in DC characteristic.
And AC characteristic is guaranteed when more than 3.0 V.
When the power supply voltage is VPOC to 3.0 V, the device does not malfunction.

Note 4. Reliability restrictions from 1.35 V to 1.43 V

### ADVANCED, PREMIUM

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| CPU clock frequency | $f_{CPUCLK}$ | | | | 120 | MHz |
| | $f_{CPUCLK2}$ | for OSTMn | | | 60 | MHz |
| Peripheral clock (clock domain) frequency*1 | $f_{CKSCLK_AWDTA}$ | for WDTA0 | | | 240*2 | kHz |
| | $f_{CKSCLK_ATAUJ}$ | for TAUJ0 | | | 40 | MHz |
| | $f_{CKSCLK_ARTCA}$ | for RTCA0 | | | 4 | MHz |
| | $f_{CKSCLK_AADCA}$ | for ADCA0 | | | 40 | MHz |
| | $f_{CKSCLK_AFOUT}$ | for FOUT | | | 24 | MHz |
| | $f_{CKSCLK_IPERI1}$ | for TAUD0 | | | 80 | MHz |
| | | for TAUJ1 | | | | |
| | | for ENCA0 | | | | |
| | | for TAPA0 | | | | |
| | | for PIC0 | | | | |
| | $f_{CKSCLK_IPERI2}$ | for TAUBn | | | 40 | MHz |
| | | for RS-CANn (clkc) | | | | |
| | | for PWBAn | | | | |
| | | for PWGAn | | | | |
| | | for PWSAn | | | | |
| | $f_{CKSCLK_ILIN}$ | for RLIN2m | | | 40 | MHz |
| | | for RLIN3n | | | | |
| | $f_{CKSCLK_IADCA}$ | for ADCA1 | | | 40 | MHz |
| | $f_{CKSCLK_ICANOSC}$ | for RS-CANn (clk_xincan) | | | 24 | MHz |
| | $f_{CKSCLK_ICSI}$ | for CSIGn | | | 80 | MHz |
| | | for CSIHn | | | | |
| | $f_{LS\ IntOSC}$ | for WDTA1 | | | 240*2 | kHz |
| | $f_{CKSCLK_IIIC}$ | for RIICn | | | 40 | MHz |
| | $f_{EMCLK}$ | for LPSn | | | 8 | MHz |
| Power supply | REGVCC | REGVCC = EVCC | VPOC*3 | | 5.5 | V |
| | EVCC | | | | | |
| | A0VREF | | 3.0 | | 5.5 | V |
| | A1VREF | | | | | |
| Normal operation voltage | AWOVCL | | 1.1 | 1.25 | 1.35 | V |
| | ISOVCL | | | | | |
| Limited operation voltage*4 | AWOVCL | | 1.35 | | 1.43 | V |
| | ISOVCL | | | | | |

Note 1. For clock specification of peripherals, see **Section 12, Clock Controller**.

Note 2. This frequency depends on the internal oscillator (LS IntOSC).

Note 3. “VPOC” means POC (power on clear) detection voltage (TYP. 2.85 V). For detail, see **Section 40.8.2, Voltage Detector (POC, LVI, VLVI, CVM) Characteristics**.
In addition, the guaranteed operation in DC characteristic.
And AC characteristic is guaranteed when more than 3.0 V.
When the power supply voltage is VPOC to 3.0 V, the device does not malfunction.

Note 4. Reliability restrictions from 1.35 V to 1.43 V

## 40.5 Oscillator Characteristics

**Condition:** REGVCC = EVCC = VPOC to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| MainOSC frequency*4 | $f_{MOSC}$ | Crystal/Ceramic | 16 | 20 | 24 | MHz |
| MainOSC current consumption | $I_{MOSC}$ | Crystal/Ceramic After stabilization | | 1.9*3 | 2.3*3 | mA |
| MainOSC oscillation start point | $V_{MOSCSP}$ | Crystal/Ceramic | VPOC | | | V |
| MainOSC oscillation operating point | $V_{MOSCOP}$ | Crystal/Ceramic | | 0.5 × REGVCC*3 | | V |
| MainOSC oscillation amplitude | $V_{MOSCAMP}$ | Crystal/Ceramic | 0.4 × REGVCC – 0.2*3 | | | V |
| MainOSC oscillation stabilization time | $t_{MSTB}$ | Crystal/Ceramic | | 2 *1, *3 | | ms |
| X1 clock Input frequency*4 | $f_{EX}$ | | 16 | 20 | 24 | MHz |
| X1 clock Input cycle time | $t_{EXCYC}$ | | 41.7 | 50 | 62.5 | ns |
| X1 High level Input voltage | $V_{IH}$ | | 0.7 × REGVCC | | REGVCC + 0.5 | V |
| | | @Flash Programming Interface*5 | 0.8 × REGVCC | | REGVCC + 0.5 | V |
| X1 Low level Input voltage | $V_{IL}$ | | –0.5 | | 0.3 × REGVCC | V |
| | | @Flash Programming Interface*5 | –0.5 | | 0.2 × REGVCC | V |
| X1 Input leakage current | $I_{LIH}$ | VI = REGVCC | | | 0.5 | µA |
| | $I_{LIL}$ | VI = 0 V | | | –0.5 | µA |
| X1 clock Input low-level pulse width | $t_{EXL}$ | $f_{EX}$=16MHz | 26 | | | ns |
| | | $f_{EX}$=20MHz | 20 | | | ns |
| | | $f_{EX}$=24MHz | 16 | | | ns |
| X1 clock Input high-level pulse width | $t_{EXH}$ | $f_{EX}$=16MHz | 26 | | | ns |
| | | $f_{EX}$=20MHz | 20 | | | ns |
| | | $f_{EX}$=24MHz | 16 | | | ns |
| X1 clock Input period jitter | | | –0.3 | | 0.3 | ns |
| SubOSC frequency | $f_{SOSC}$ | Crystal | 30 | 32.768 | 38 | kHz |
| SubOSC current consumption | $I_{SOSC}$ | After stabilization | | 1.5*3 | 4*3 | µA |
| SubOSC DC operating point | $V_{SOSCDCOP}$ | | | 0.65*3 | | V |
| SubOSC oscillation stabilization time | $t_{SSTB}$ | | | *2 | | s |

Note 1. Oscillator stabilization time is time until being set (“1”) in MOSCS.MOSCCLKACT bit after MOSCE.MOSCENTRG bit is written “1”, and depends on the setting value of MOSCST register. Please decide appropriate oscillation stabilization time by matching test with resonator and oscillation circuit.

Note 2. Oscillator stabilization time is time until being set (“1”) in SOSCS.SOSCCLKACT bit after SOSCE.SOSCENTRG bit is written “1”, and depends on the setting value of SOSCST register. Please decide appropriate oscillation stabilization time by matching test with resonator and oscillation circuit.

Note 3. This is reference value.

Note 4. The following three crystal/ceramic resonator frequencies are supported: 16 MHz, 20 MHz and 24 MHz.

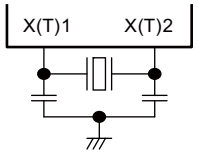
Note 5. X2 should be open and its parasitic capacitance should be less than 5pF.

**CAUTION**

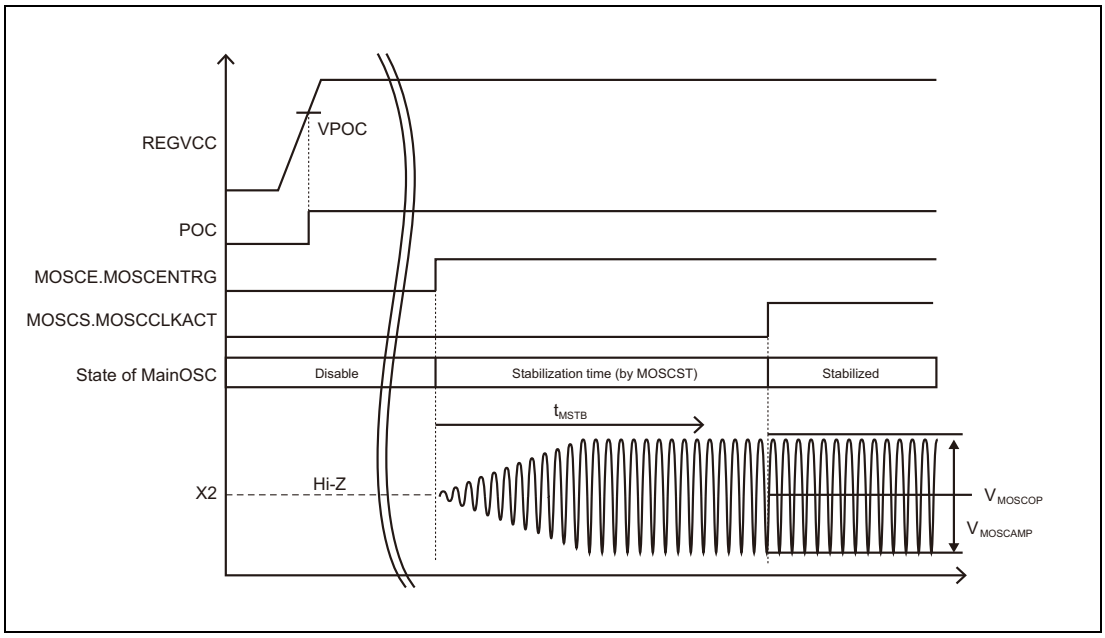
The oscillation stabilization time differs according the matching with the external resonator circuit. It is recommended to determine the oscillation stabilization time by an oscillator matching test.

**NOTE**

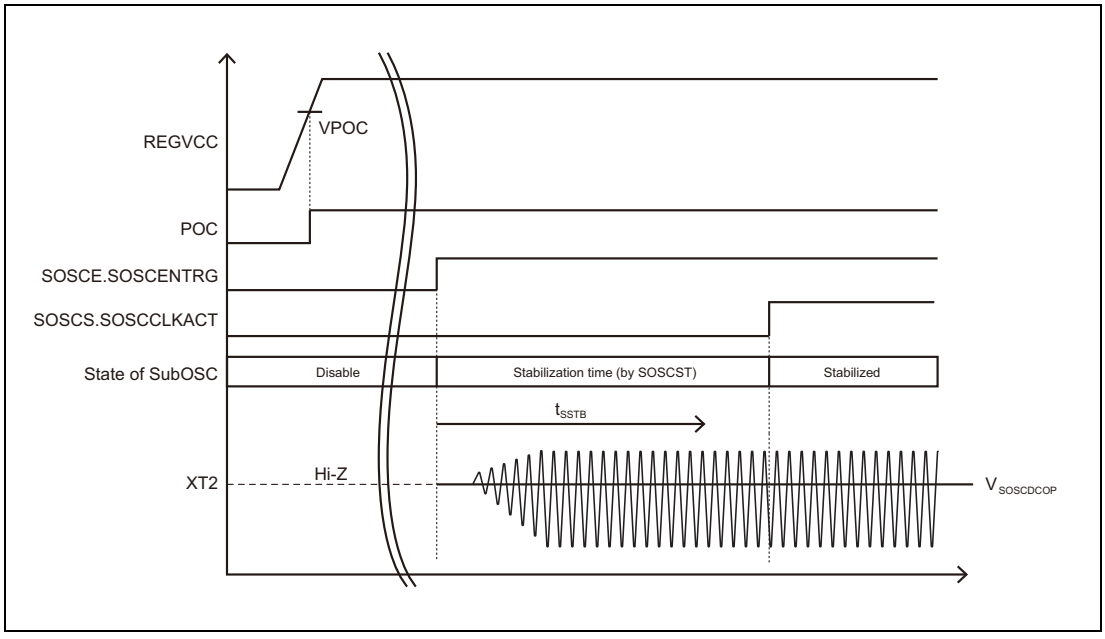
Recommended oscillator circuit is shown below.



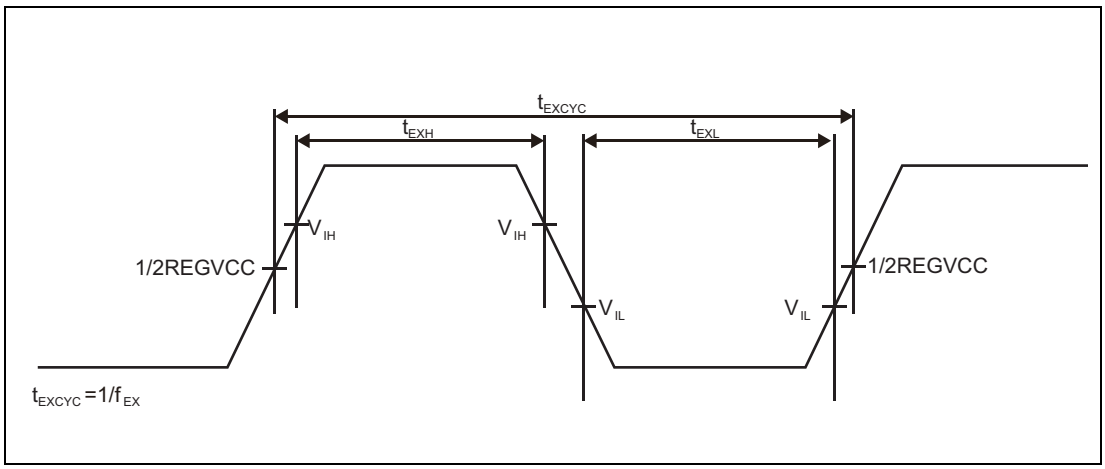

MainOSC




SubOSC



External clock




## 40.6 Internal Oscillator Characteristics

**Condition:** REGVCC = EVCC = VPOC to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| LS IntOSC frequency | $f_{RL}$ | | 220.8 | 240 | 259.2 | kHz |
| HS IntOSC frequency | $f_{RH}$ | | 7.6 | 8 | 8.4 | MHz |
| | | After user trimming @ trimming temp | 7.92 | 8 | 8.08 | MHz |
| HS IntOSC current consumption | $I_{RH}$ | After stabilization | | | 170*1 | µA |
| HS IntOSC oscillation stabilization time | $t_{RHSTB}$ | | | | 54.4 | µs |

Note 1. This is reference value.

## 40.7 PLL Characteristics

**Condition:** REGVCC = EVCC = VPOC to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Input frequency | $f_{PLLICLK}$ | MainOSC | 16 | 20 | 24 | MHz |
| | | HS IntOSC After user trimming @ trimming temp*3 | 7.92 | 8.0 | 8.08 | MHz |
| Output frequency (PLL for CPU) | $f_{CPLL}$ | ECO | 63.3 | | 80 | MHz |
| | | ADVANCED, PREMIUM | 63.3 | | 120 | MHz |
| Output frequency (PLL for Peripheral) | $f_{PPLL}$ | | 63.3 | | 80 | MHz |
| Output period jitter*1 | $t_{CPJ}$ | | –100 | | 100 | ps |
| Long term jitter*1 | $t_{LTJ}$ | term = 1 µs | –500 | | 500 | ps |
| | | term = 10 µs | –1 | | 1 | ns |
| | | term = 20 µs | –2 | | 2 | ns |
| Lock time*2 | $t_{LCKP}$ | | 104 | 112.3 | 122.1 | µs |

Note 1. This is reference value.

Note 2. Lock time is time until being set (“1”) in PLLS.PLLCLKACT bit after PLLE.PLLENTRG bit is written “1”.

Note 3. HS IntOSC supports $F_{CPLL}$ = 80 MHz only. Please use it after an evaluation in customers enough.

# 40.8 Power Management Characteristics

## 40.8.1 Regulator Characteristics

**Condition:** REGVCC = EVCC = VPOC to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
Tj = –40 to (depend on the product) °C, CL = 30 pF

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Input voltage | REGVCC | | VPOC*1 | | 5.5 | V |
| Output voltage | AWOVCL | AWOVCL pin | 1.15 | 1.25 | 1.35 | V |
| | ISOVCL | ISOVCL pin | 1.15 | 1.25 | 1.35 | V |
| Capacitance | CAWOVCL | AWOVCL pin | 0.07 | 0.10 | 0.13 | µF |
| | CISOVCL | ISOVCL pin | 0.07 | 0.10 | 0.13 | µF |
| Equivalent series resistance for load capacitance | RVRAWO | for CAWOVCL | | | 40*2 | mΩ |
| | RVRISO | for CISOVCL | | | 40*2 | mΩ |
| Inrush current during power-on | | | | | 120 | mA |

Note 1. “VPOC” means POC (power on clear) detection voltage (typ. 2.85V). For detail, see **Section 40.8.2, Voltage Detector (POC, LVI, VLVI, CVM) Characteristics**.

Note 2. This is reference value.

## 40.8.2 Voltage Detector (POC, LVI, VLVI, CVM) Characteristics

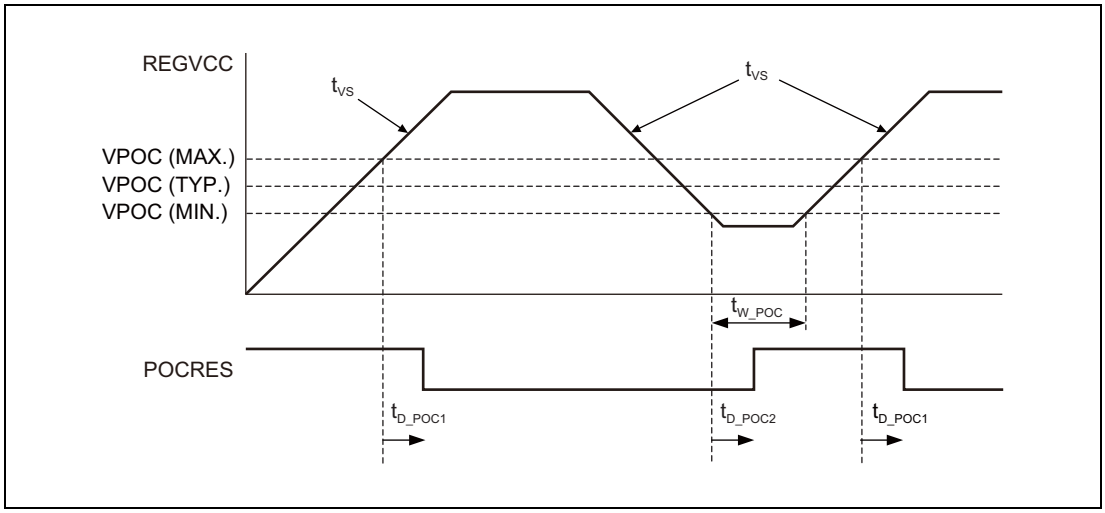
**Condition:** REGVCC = EVCC = VPOC to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

| Item | Symbol | Condition | | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|---|
| Detection voltage (REGVCC) | VPOC | POC | | | 2.7 | 2.85 | 3.0 | V |
| | VLVI0 | LVI | Rise | | 3.87 | 4.0 | 4.13 | V |
| | | | Fall | | 3.9 | 4.0 | 4.1 | V |
| | VLVI1 | | Rise | | 3.57 | 3.7 | 3.83 | V |
| | | | Fall | | 3.6 | 3.7 | 3.8 | V |
| | VLVI2 | | Rise | | 3.37 | 3.5 | 3.63 | V |
| | | | Fall | | 3.4 | 3.5 | 3.6 | V |
| | VVLVI | VLVI | | | 1.8 | 1.9 | 2.0 | V |
| Detection voltage (ISOVCL) | VCVMH | CVM | High voltage**Caution** | | 1.35 | 1.39 | 1.43 | V |
| | VCVML*8 | | Low voltage**Caution** | | 1.10 | 1.15 | 1.20 | V |
| Response time | $t_{D_POC1}$*6 | POC | At power-on (Rise) | *1 | | | 2 | ms |
| | | | | *2 | | | 6.3 | ms |
| | | | After power-on (Rise) | *3 | | | 2 | ms |
| | | | | *4 | | | 5 | ms |
| | $t_{D_POC2}$*7 | | After power-on (Fall) | *5 | | | 5 | µs |
| | $t_{D_LVI}$ | LVI | | | | | 2 | ms |
| | $t_{D_VLVI}$ | VLVI | | *3 | | | 2 | ms |
| | | | | *4 | | | 5 | ms |
| | $t_{D_CVM}$ | CVM | | | 0.2 | | 10 | µs |
| Setup time | $t_{S_LVI}$ | LVI | LVICNT0,1 bits are set to 1 (except $00_B$), then LVI is ready to operate | | | | 80 | µs |
| REGVCC minimum width | $t_{W_POC}$ | POC | | | 0.2 | | | ms |
| | $t_{W_LVI}$ | LVI | | | 0.2 | | | ms |
| | $t_{W_VLVI}$ | VLVI | | | 0.2 | | | ms |

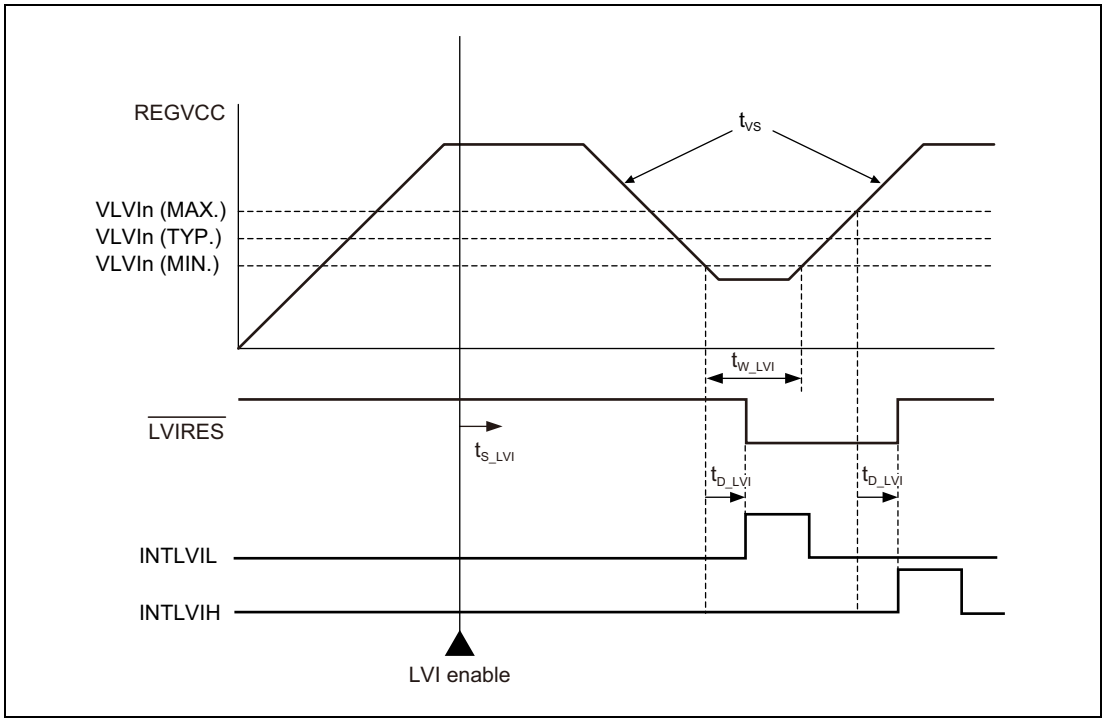
Note 1. Voltage slope ($t_{VS}$) : 0.02 V/ms ≤ $t_{VS}$ ≤ 0.5 V/ms
Note 2. Voltage slope ($t_{VS}$) : 0.5 V/ms < $t_{VS}$ ≤ 500 V/ms
Note 3. Voltage slope ($t_{VS}$) : 0.02 V/ms ≤ $t_{VS}$ ≤ 20 V/ms
Note 4. Voltage slope ($t_{VS}$) : 20 V/ms < $t_{VS}$ ≤ 500 V/ms
Note 5. Voltage slope ($t_{VS}$) : 0.02 V/ms ≤ $t_{VS}$ ≤ 500 V/ms
Note 6. $t_{D_POC1}$ is the time from detection voltage to release of reset signal.
Note 7. $t_{D_POC2}$ is the time from detection voltage to occurrence of reset signal.
Note 8. The CVM monitors the internal voltage regulator output to ensure that ISOVCL is upper than specified minimum level.

**Caution:** A detection of the voltage ISOVCL outside the specified level of VCVMH and VCVML is not ensured by CVM.

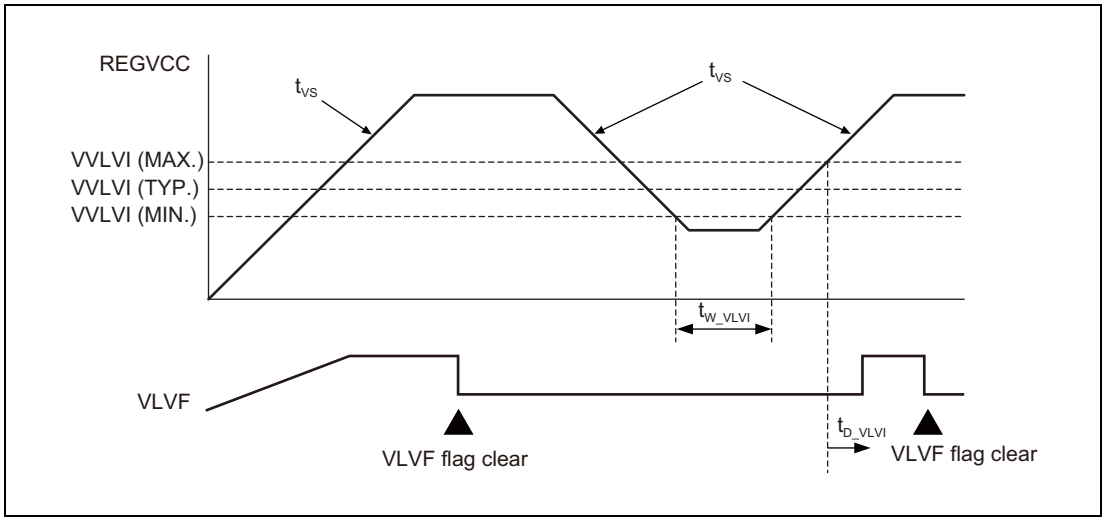
<POC>


REGVCC
VPOC (MAX.)
VPOC (TYP.)
VPOC (MIN.)
tVS
tVS
tW_POC
POCRES
tD_POC1
tD_POC2
tD_POC1

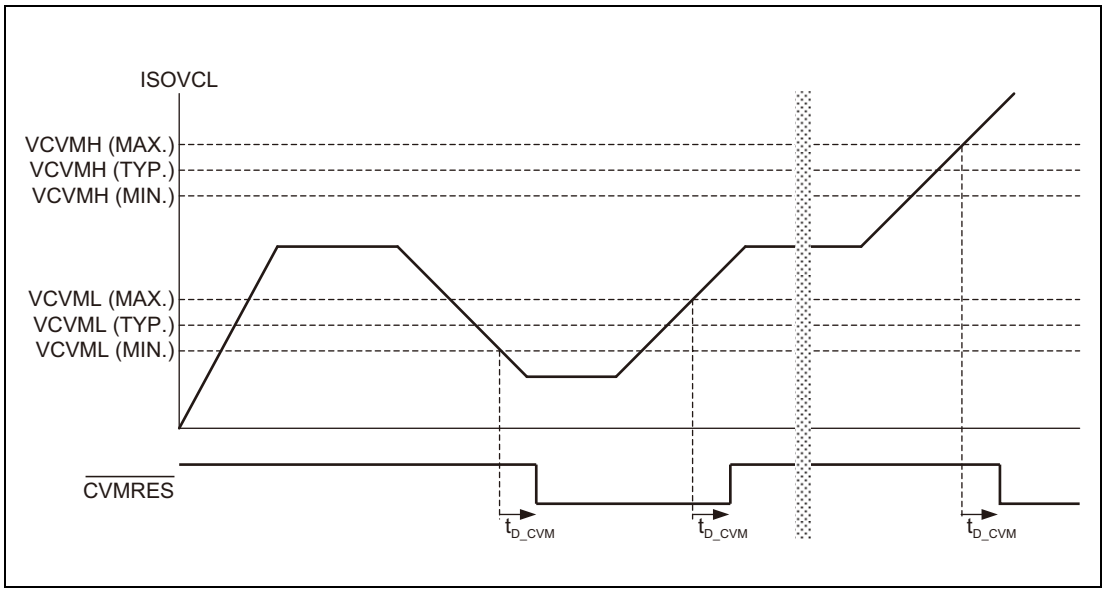
<LVI>


REGVCC
VLVIn (MAX.)
VLVIn (TYP.)
VLVIn (MIN.)
tVS
tW_LVI
LVIRES
tS_LVI
tD_LVI
tD_LVI
INTLVIL
INTLVIH
LVI enable

<VLVI>


REGVCC
$t_{VS}$
$t_{VS}$
VVLVI (MAX.)
VVLVI (TYP.)
VVLVI (MIN.)
$t_{W_VLVI}$
VLVF
VLVF flag clear
$t_{D_VLVI}$
VLVF flag clear

<CVM>

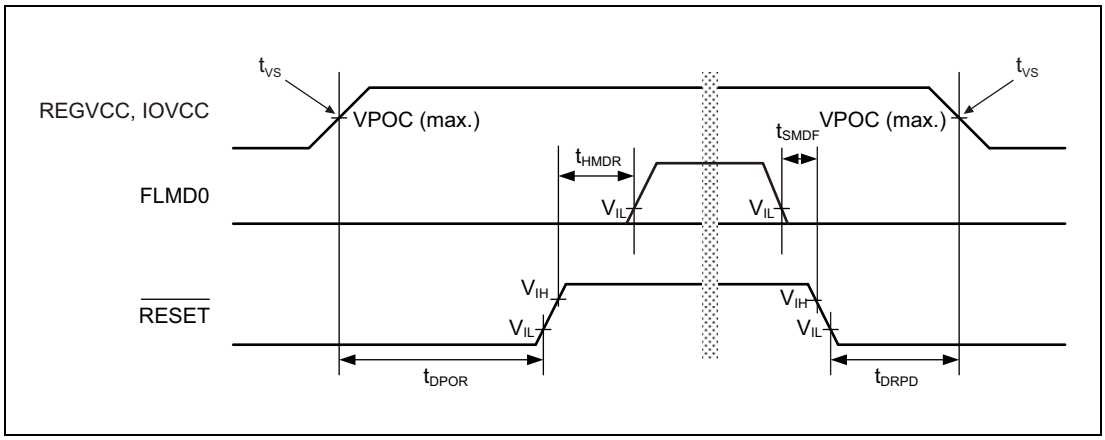

ISOVCL
VCVMH (MAX.)
VCVMH (TYP.)
VCVMH (MIN.)
VCVML (MAX.)
VCVML (TYP.)
VCVML (MIN.)
CVMRES
$t_{D_CVM}$
$t_{D_CVM}$
$t_{D_CVM}$

## 40.8.3 Power Up/Down Timing

**Condition:** REGVCC = EVCC = VPOC to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

**Table 40.1 In case the $\overline{\text{RESET}}$ pin is used (for Normal Operation Mode)**

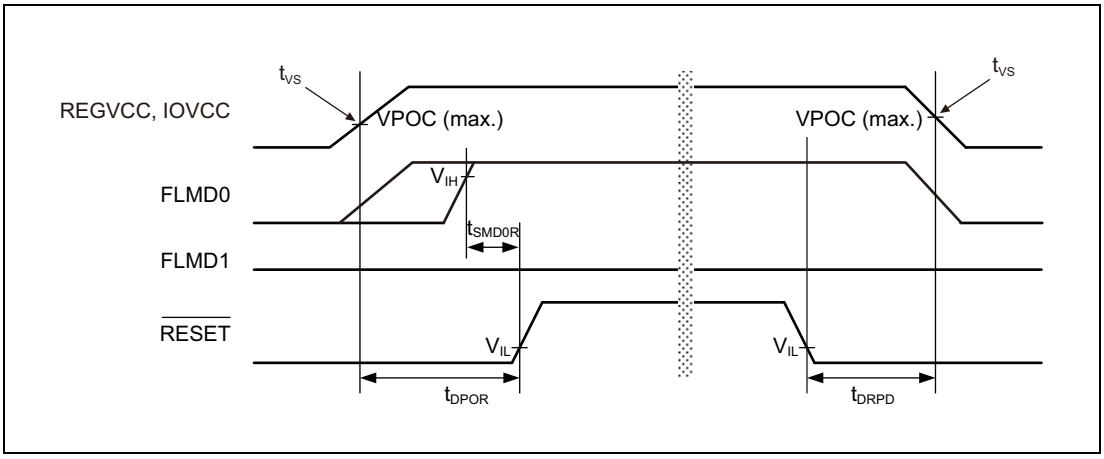
| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Voltage slope (REGVCC and IOVCC*1) | $t_{VS}$ | | 0.02 (= 50 ms/V) | | 500 (= 2 µs/V) | V/ms |
| REGVCC ↑ and IOVCC*1 ↑ to $\overline{\text{RESET}}$ ↑ delay time | $t_{DPOR}$ | Voltage slope ($t_{VS}$) : 0.02 V/ms ≤ $t_{VS}$ ≤ 0.5 V/ms | 2 | | | ms |
| | | Voltage slope ($t_{VS}$) : 0.5 V/ms < $t_{VS}$ ≤ 500 V/ms | 6.3 | | | ms |
| FLMD0 hold time (vs $\overline{\text{RESET}}$ ↑) | $t_{HMDR}$ | | 1 | | | ms |
| FLMD0 setup time (vs $\overline{\text{RESET}}$ ↓) | $t_{SMDF}$ | | 0 | | | µs |
| $\overline{\text{RESET}}$ ↓ to REGVCC ↓ and IOVCC*1 ↓ delay time | $t_{DRPD}$ | | 0 | | | ms |




Note 1. IOVCC means EVCC, A0VREF and A1VREF.

**Table 40.2 In case the $\overline{\text{RESET}}$ pin is used (for Serial programming mode)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Voltage slope (REGVCC and IOVCC*1) | $t_{VS}$ | | 0.02 (= 50 ms/V) | | 500 (= 2 µs/V) | V/ms |
| REGVCC ↑ and IOVCC*1 ↑ to $\overline{\text{RESET}}$ ↑ delay time | $t_{DPOR}$ | Voltage slope ($t_{VS}$) : 0.02 V/ms ≤ $t_{VS}$ ≤ 0.5 V/ms | 2 | | | ms |
| | | Voltage slope ($t_{VS}$) : 0.5 V/ms < $t_{VS}$ ≤ 500 V/ms | 6.3 | | | ms |
| FLMD0 setup time (vs $\overline{\text{RESET}}$ ↑) | $t_{SMD0R}$ | | 1 | | | ms |
| $\overline{\text{RESET}}$ ↓ to REGVCC ↓ and IOVCC*1 ↓ delay time | $t_{DRPD}$ | | 0 | | | ms |




Note 1. IOVCC means EVCC, A0VREF and A1VREF.

**Table 40.3 In case the $\overline{\text{RESET}}$ pin is used (for Boundary scan mode)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Voltage slope (REGVCC and IOVCC*1) | $t_{VS}$ | | 0.02 (= 50 ms/V) | | 500 (= 2 µs/V) | V/ms |
| $\overline{\text{REGVCC}}$ ↑ and IOVCC*1 ↑ to $\overline{\text{RESET}}$ ↑ delay time | $t_{DPOR}$ | Voltage slope ($t_{VS}$) : 0.02 V/ms ≤ $t_{VS}$ ≤ 0.5 V/ms | 2 | | | ms |
| | | Voltage slope ($t_{VS}$) : 0.5 V/ms < $t_{VS}$ ≤ 500 V/ms | 6.3 | | | ms |
| FLMD0 setup time (vs $\overline{\text{RESET}}$ ↑) | $t_{SMD0R}$ | | 1 | | | ms |
| FLMD1, MODE0, MODE1 setup time (vs FLMD0 ↑) | $t_{SMD1R}$ | | 1 | | | µs |
| FLMD0 hold time (vs $\overline{\text{RESET}}$ ↓) | $t_{HMD0F}$ | | 1 | | | µs |
| FLMD1, MODE0, MODE1, MODE2 hold time (vs FLMD0 ↓) | $t_{HMD1F}$ | | 1 | | | µs |
| $\overline{\text{RESET}}$↓ to REGVCC ↓ and IOVCC*1 ↓ delay time | $t_{DRPD}$ | | 0 | | | ms |
| $\overline{\text{DCUTRST}}$ input delay time (vs $\overline{\text{RESET}}$ ↑) | $t_{DRTRST}$ | | 1 | | | ms |
| $\overline{\text{RESET}}$ hold time (vs $\overline{\text{DCUTRST}}$ ↓) | $t_{HRTRST}$ | | 0 | | | ms |

Note 1. IOVCC means EVCC, A0VREF and A1VREF.

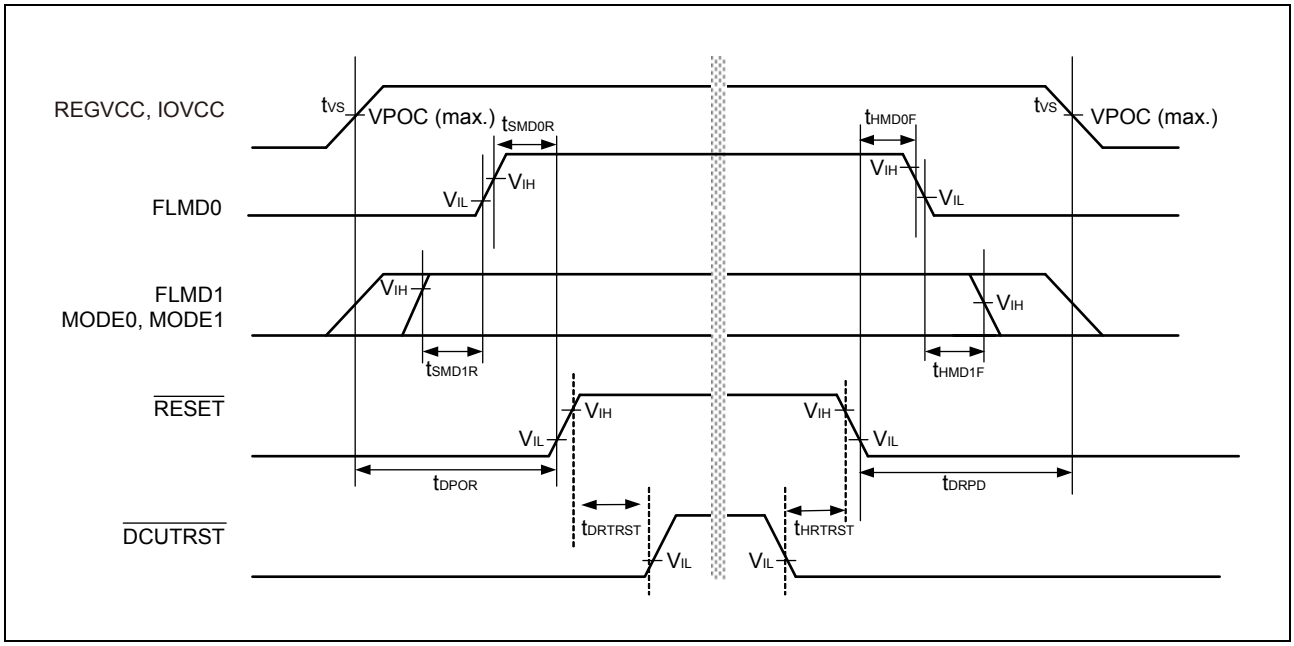


**Table 40.4 In case the $\overline{\text{RESET}}$ pin is used (for User boot mode)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Voltage slope (REGVCC and IOVCC*1) | $t_{VS}$ | | 0.02 (= 50 ms/V) | | 500 (= 2 µs/V) | V/ms |
| REGVCC ↑ and IOVCC*1 ↑ to $\overline{\text{RESET}}$ ↑ delay time | $t_{DPOR}$ | Voltage slope ($t_{VS}$) : 0.02 V/ms ≤ $t_{VS}$ ≤ 0.5 V/ms | 2 | | | ms |
| | | Voltage slope ($t_{VS}$) : 0.5 V/ms < $t_{VS}$ ≤ 500 V/ms | 6.3 | | | ms |
| FLMD0 setup time (vs $\overline{\text{RESET}}$ ↑) | $t_{SMD0R}$ | | 1 | | | ms |
| FLMD1, MODE0, MODE1, MODE2 setup time (vs FLMD0 ↑) | $t_{SMD1R}$ | | 1 | | | µs |
| FLMD0 hold time (vs $\overline{\text{RESET}}$ ↓) | $t_{HMD0F}$ | | 1 | | | µs |
| FLMD1, MODE0, MODE1, MODE2 hold time (vs FLMD0 ↓) | $t_{HMD1F}$ | | 1 | | | µs |
| $\overline{\text{RESET}}$ ↓ to REGVCC ↓ and IOVCC*1 ↓ delay time | $t_{DRPD}$ | | 0 | | | ms |

Note 1. IOVCC means EVCC, A0VREF and A1VREF.

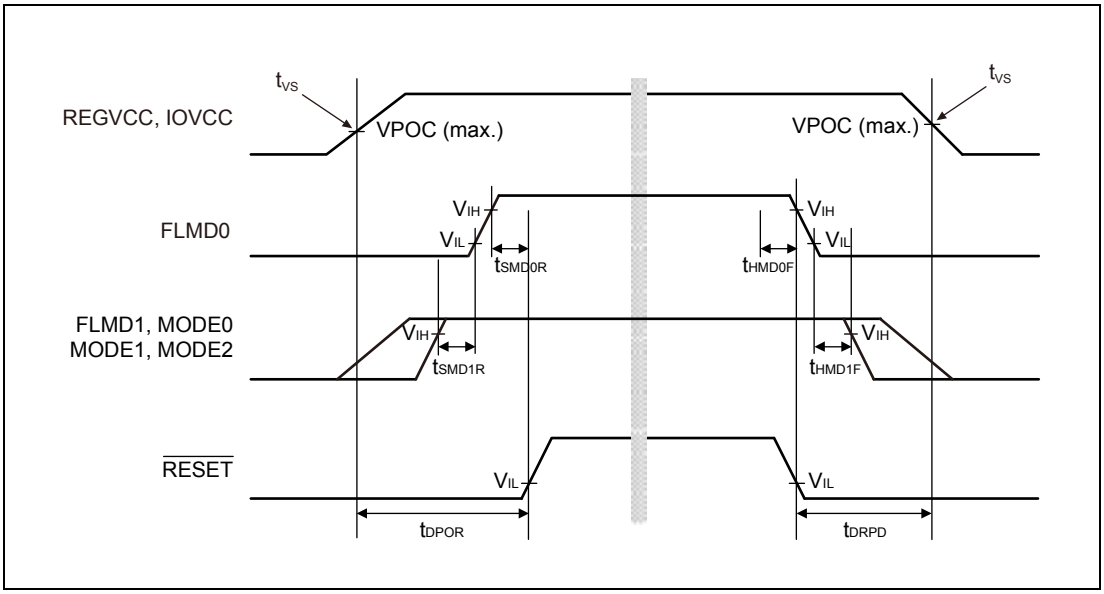
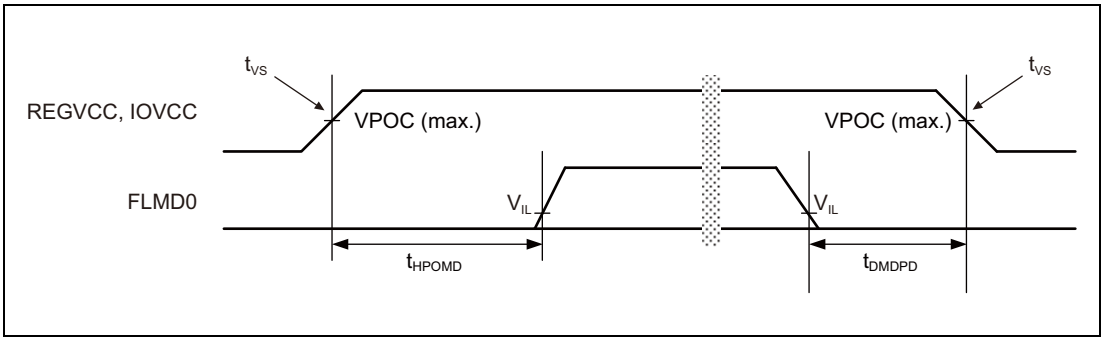


**Table 40.5 In case the $\overline{\text{RESET}}$ pin is not used and fixed to high level by pull-up*1**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Voltage slope (REGVCC and IOVCC*2) | $t_{VS}$ | | 0.02 (= 50 ms/V) | | 500 (= 2 µs/V) | V/ms |
| REGVCC ↑ and IOVCC*2 ↑ to FLMD0 hold time | $t_{HPOMD}$ | Voltage slope ($t_{VS}$) : 0.02 V/ms ≤ $t_{VS}$ ≤ 0.5 V/ms | 2 | | | ms |
| | | Voltage slope ($t_{VS}$) : 0.5 V/ms < $t_{VS}$ ≤ 500 V/ms | 6.3 | | | ms |
| FLMD0 ↓ to REGVCC ↓ and IOVCC*2 ↓ delay time | $t_{DMDPD}$ | | 1 | | | µs |

Note 1. This operating condition is available only in normal operation mode (include self-programming mode). When the device is used in except normal operation mode, please use the $\overline{\text{RESET}}$ pin.

Note 2. IOVCC means EVCC, A0VREF and A1VREF.



### 40.8.4 CPU Reset Release Timing

**Condition:** REGVCC = EVCC = VPOC to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

**Table 40.6 In case the $\overline{\text{RESET}}$ pin is not used**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| REGVCC ↑ to CPU reset release*1 | $t_{DPCRR}$ | Voltage slope ($t_{VS}$) : 0.02 V/ms ≤ $t_{VS}$ ≤ 0.5 V/ms | | | 2.58 | ms |
| | | Voltage slope ($t_{VS}$) : 0.5 V/ms < $t_{VS}$ ≤ 500 V/ms | | | 8.3 | ms |

Note 1. This is reference value.

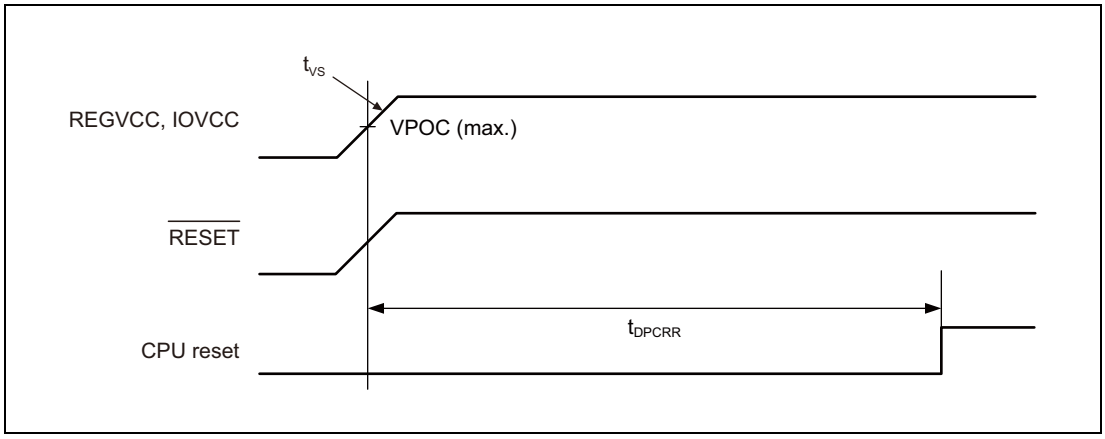
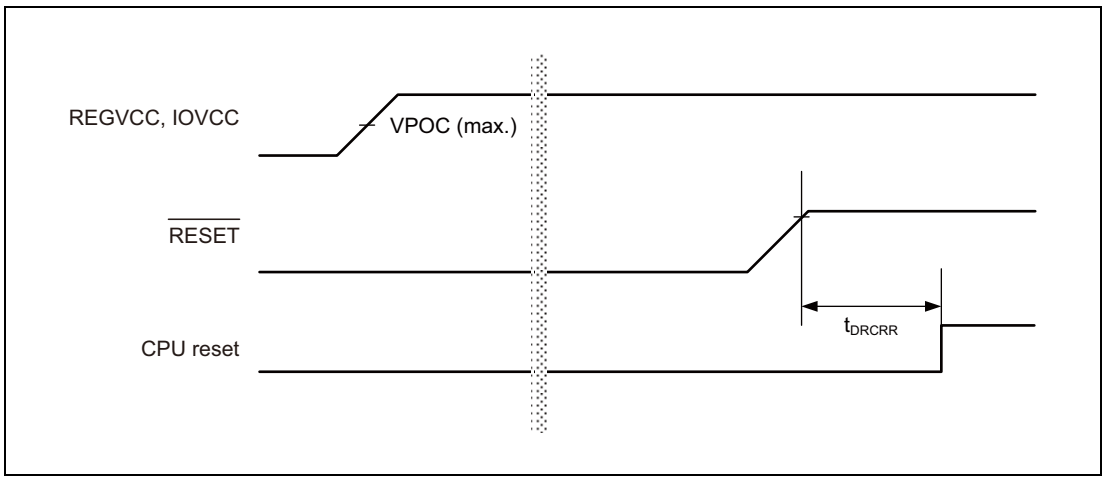



**Table 40.7 In case the $\overline{\text{RESET}}$ pin is used**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{RESET}}$ ↑ to CPU reset release*1 | $t_{DRCRR}$ | | | | 16*2 | µs |

Note 1. This is reference value.

Note 2. At least $t_{DPCRR}$ time is necessary reaching from VPOC (max) even if power up sequence is kept shown on **Section 40.8.3, Power Up/Down Timing**.



## 40.9 Pin Characteristics

**Condition:** Some of the conditions mentioned in this chapter can be selected by software and described in the hardware user's manual.

(1/4)

| | Port Input Buffer Function | | | | | | Port Output Drive Strength Mode | Other Port Function | |
|---|---|---|---|---|---|---|---|---|---|
| Pin Name | CMOS | SHMT1 | SHMT2 | SHMT4 | TTL | Analog | | Pull-up | Pull-down |
| $\overline{\text{RESET}}$ | — | — | √ | — | — | — | — | — | — |
| FLMD0 | — | √ | — | — | — | — | — | √ | √ |
| AP0_0 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP0_1 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP0_2 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP0_3 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP0_4 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP0_5 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP0_6 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP0_7 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP0_8 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP0_9 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP0_10 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP0_11 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP0_12 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP0_13 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP0_14 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP0_15 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP1_0 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP1_1 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP1_2 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP1_3 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP1_4 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP1_5 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP1_6 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP1_7 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP1_8 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP1_9 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP1_10 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP1_11 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP1_12 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP1_13 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP1_14 | √ | — | — | — | — | √ | Slow | — | √*1 |
| AP1_15 | √ | — | — | — | — | √ | Slow | — | √*1 |
| IP0_0 | — | — | — | — | — | — | — | — | — |
| JP0_0 | — | √ | — | √ | √ | — | Slow | √ | √ |
| JP0_1 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| JP0_2 | — | √ | — | √ | √ | — | Slow/Fast | √ | √ |
| JP0_3 | — | √ | — | √ | √ | — | Slow/Fast | √ | √ |
| JP0_4 | — | — | — | √ | √*5 | — | Slow | √ | √ |
| JP0_5 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| JP0_6 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P0_0 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |

(2/4)

| | Port Input Buffer Function | | | | | | Port Output Drive Strength Mode | Other Port Function | |
|---|---|---|---|---|---|---|---|---|---|
| Pin Name | CMOS | SHMT1 | SHMT2 | SHMT4 | TTL | Analog | | Pull-up | Pull-down |
| P0_1 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P0_2 | — | √ | — | √ | — | — | Slow/Fast*2 | √ | √ |
| P0_3 | — | √ | — | √ | — | — | Slow/Fast*2 | √ | √ |
| P0_4 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P0_5 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P0_6 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P0_7 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P0_8 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P0_9 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P0_10 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P0_11 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P0_12 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P0_13 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P0_14 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P1_0 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P1_1 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P1_2 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P1_3 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P1_4 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P1_5 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P1_6 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P1_7 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P1_8 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P1_9 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P1_10 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P1_11 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P1_12 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P1_13 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P1_14 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P1_15 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P10_0 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P10_1 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P10_2 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P10_3 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P10_4 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P10_5 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P10_6 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P10_7 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P10_8 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P10_9 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P10_10 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P10_11 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P10_12 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P10_13 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P10_14 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P10_15 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |

(3/4)

| | Port Input Buffer Function | | | | | | Port Output Drive Strength Mode | Other Port Function | |
|---|---|---|---|---|---|---|---|---|---|
| Pin Name | CMOS | SHMT1 | SHMT2 | SHMT4 | TTL | Analog | | Pull-up | Pull-down |
| P11_0 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P11_1 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P11_2 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P11_3 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P11_4 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P11_5 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P11_6 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P11_7 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P11_8 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P11_9 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P11_10 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P11_11 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P11_12 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P11_13 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P11_14 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P11_15 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P12_0 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P12_1 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P12_2 | — | √ | — | √ | — | — | Slow/Fast*3 | √ | √ |
| P12_3 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P12_4 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P12_5 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P18_0 | — | √ | — | √ | — | √ | Slow/Fast*3 | √ | √*4 |
| P18_1 | — | √ | — | √ | — | √ | Slow/Fast*3 | √ | √*4 |
| P18_2 | — | √ | — | √ | — | √ | Slow/Fast*3 | √ | √*4 |
| P18_3 | — | √ | — | √ | — | √ | Slow/Fast*3 | √ | √*4 |
| P18_4 | — | √ | — | √ | — | √ | Slow/Fast*3 | √ | √*4 |
| P18_5 | — | √ | — | √ | — | √ | Slow/Fast*3 | √ | √*4 |
| P18_6 | — | √ | — | √ | — | √ | Slow/Fast*3 | √ | √*4 |
| P18_7 | — | √ | — | √ | — | √ | Slow/Fast*3 | √ | √*4 |
| P2_0 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P2_1 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P2_2 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P2_3 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P2_4 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P2_5 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P2_6 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P20_0 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P20_1 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P20_2 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P20_3 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P20_4 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P20_5 | — | √ | — | √ | — | — | Slow/Fast | √ | √ |
| P8_0 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P8_1 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P8_2 | — | √ | — | √ | — | √ | Slow | √ | √*4 |

(4/4)

| | Port Input Buffer Function | | | | | | Port Output Drive Strength Mode | Other Port Function | |
|---|---|---|---|---|---|---|---|---|---|
| Pin Name | CMOS | SHMT1 | SHMT2 | SHMT4 | TTL | Analog | | Pull-up | Pull-down |
| P8_3 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P8_4 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P8_5 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P8_6 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P8_7 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P8_8 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P8_9 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P8_10 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P8_11 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P8_12 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P9_0 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P9_1 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P9_2 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P9_3 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P9_4 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P9_5 | — | √ | — | √ | — | √ | Slow | √ | √*4 |
| P9_6 | — | √ | — | √ | — | √ | Slow | √ | √*4 |

Note 1. Pull-down resistor for ADC diagnostic purpose. Control via ADC self-diagnostic register.

Note 2. Supports Cload: 100pF

Note 3. Supports Cload: 50pF

Note 4. Pull-down resistors for ADC diagnostic and internal pull-down purposes. For ADC diagnostic, control via ADC self-diagnostic register. For internal pull-down, control via PD register.

Note 5. TTL is selected for Boundary scan mode or Nexus in normal operating mode.

**Condition:** REGVCC = EVCC = VPOC to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

(1/2)

| Item | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| High level input voltage | VIH | CMOS | | 0.65 ×IOVCC | | IOVCC + 0.3 | V |
| | | SHMT1*3 | | 0.65 × IOVCC | | IOVCC + 0.3 | V |
| | | SHMT2 | | 0.75 × IOVCC | | IOVCC + 0.3 | V |
| | | SHMT4 | | 0.8 × IOVCC | | IOVCC + 0.3 | V |
| | | TTL | EVCC = VPOC to 3.6 V | 2.0 | | IOVCC + 0.3 | V |
| | | | EVCC = 3.6 V to 5.5 V | 2.2 | | IOVCC + 0.3 | V |
| | | IP0_0 pin | | 0.7 × REGVCC | | REGVCC | V |
| Low level input voltage | VIL | CMOS | | –0.3 | | 0.35 × IOVCC | V |
| | | SHMT1 | | –0.3 | | 0.35 × IOVCC | V |
| | | SHMT2 | | –0.3 | | 0.25 × IOVCC | V |
| | | SHMT4 | | –0.3 | | 0.5 × IOVCC | V |
| | | TTL | | –0.3 | | 0.8 | V |
| | | IP0_0 pin | | 0 | | 0.3 × REGVCC | V |
| Input hysteresis width | VH | SHMT1 | | 0.3 | | | V |
| | | SHMT2 | | 0.2 × IOVCC | | | V |
| | | SHMT4 | | 0.1 | | | V |
| Input leakage current | ILIH | IP0_0 pin, VI = REGVCC | | | | 0.5 | µA |
| | | $\overline{\text{RESET}}$, FLMD0, JP0, P0, P1, P2, P8, P9, P10, P11, P12, P18, P20 pin, VI = EVCC*2 | | | | 0.5 | µA |
| | | AP0 pin, VI = A0VREF*2, Tj ≤ 130°C | | | | 0.3 | µA |
| | | AP0 pin, VI = A0VREF*2 | | | | 0.5 | µA |
| | | AP1 pin, VI = A1VREF*2, Tj ≤ 130°C | | | | 0.3 | µA |
| | | AP1 pin, VI = A1VREF*2 | | | | 0.5 | µA |
| | ILIL | IP0_0 pin, VI = 0 V | | | | –0.5 | µA |
| | | $\overline{\text{RESET}}$, FLMD0, JP0, P0, P1, P2, P8, P9, P10, P11, P12, P18, P20 pin, VI = 0 V*2 | | | | –0.5 | µA |
| | | AP0 pin, VI = 0 V*2, Tj ≤ 130°C | | | | –0.3 | µA |
| | | AP0 pin, VI = 0 V*2 | | | | –0.5 | µA |
| | | AP1 pin, VI = 0 V*2, Tj ≤ 130°C | | | | –0.3 | µA |
| | | AP1 pin, VI = 0 V*2 | | | | –0.5 | µA |
| Internal pull-up resistance | RU | except FLMD0 pin, VI = 0 V | | 20 (275 µA) | 40 | 100 | kΩ |
| | | FLMD0 pin, VI = 0 V*3 | | 4 (1375 µA) | | 36 | kΩ |
| Internal pull-down resistance | RD | except FLMD0 pin, VI = EVCC | | 20 (275 µA) | 40 | 100 | kΩ |
| | | FLMD0 pin, VI = EVCC | | 4 (1375 µA) | | 36 | kΩ |
| High level output voltage | VOH | Fast mode | | | | | |
| | | | IOH = –5 mA (6 pins)*4 | IOVCC – 1.0 | | | V |
| | | | IOH = –3 mA (10 pins)*4 | IOVCC – 1.0 | | | V |
| | | | IOH = –1 mA (16 pins)*4 | IOVCC – 0.5 | | | V |
| | | | IOH = –0.1 mA (16 pins)*4 | IOVCC – 0.5 | | | V |
| | | Slow mode | | | | | |
| | | | IOH = –1 mA (16 pins)*4 | IOVCC – 0.5 | | | V |
| | | | IOH = –0.1 mA (16 pins)*4 | IOVCC – 0.5 | | | V |

(2/2)

| Item | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Low level output voltage | VOL | Fast mode | | | | | |
| | | | IOL = 5 mA (6 pins)*4 | | | 0.4 | V |
| | | | IOL = 3 mA (10 pins)*4 | | | 0.4 | V |
| | | | IOL = 1 mA (16 pins)*4 | | | 0.4 | V |
| | | Slow mode | | | | | |
| | | | IOL = 1 mA (16 pins)*4 | | | 0.4 | V |
| Rise/Fall time | $t_{KRP}/t_{KFP}$ | Fast mode (except below pins)*5 | CL = 30 pF | | | 7 | ns |
| | | | CL = 50 pF | | | 12 | ns |
| | | | CL = 100 pF | | | 24 | ns |
| | | Fast mode (P0_5, P0_6, P10_1, P10_2, P11_2, P11_3, P11_6, P11_7)*6 | CL = 50 pF | | | 6 | ns |
| | | Fast mode (P0_2, P0_3)*6 | CL = 100 pF | | | 6.15 | ns |
| | | Slow mode*5 | CL = 30 pF | | | 37 | ns |
| | | | CL = 50 pF | | | 62 | ns |
| | | | CL = 100 pF | | | 124 | ns |
| Output frequency | $f_O$ | Fast mode | CL = 30 pF | | | 40 | MHz |
| | | Slow mode | CL = 30 pF | | | 10 | MHz |
| | | | CL = 50 pF | | | 6 | MHz |
| | | | CL = 100 pF | | | 3 | MHz |

Note 1. “IOVCC” means the pins are assigned to the power supply (EVCC, A0VREF and A1VREF).

Note 2. Not select the analog input function of ADCn.

Note 3. When the internal pull-up resistor of FLMD0 pin is applied by FLMDCNT register, please connect 86 kΩ or more as external pull-down resistor.

Note 4. The number of pin indicates simultaneous ON.

Note 5. Measurement point: 0.1 × IOVCC to 0.9 × IOVCC

Note 6. Measurement point: 0.2 × IOVCC to 0.8 × IOVCC

## 40.9.1 Output Current

### 40.9.1.1 176 pin

| Item | Symbol | Pin Group | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| High-level output current | IOH | PgE | Per side1 (Total of P9_0 to P9_6, P20_0 to P20_5) | | | –30 | mA |
| | | | Per side2 (Total of P10_6 to P10_9, P18_0 to P18_7) | | | –30 | mA |
| | | | Per side3 (Total of P10_10 to P10_14, P11_1 to P11_7, P11_15, P12_0 to P12_2) | | | –30 | mA |
| | | | Per side4 (Total of P10_0 to P10_5) | | | –30 | mA |
| | | | Per side5 (Total of P0_0 to P0_3, P10_15, P11_0, P11_8 to P11_14, P12_3 to P12_5) | | | –30 | mA |
| | | | Per side6 (Total of JP0_3 to JP0_5, P0_4 to P0_6, P0_11 to P0_14, P1_0 to P1_3, P1_12, P1_13, P2_6, P8_2, P8_10 to P8_12) | | | –30 | mA |
| | | | Per side7 (Total of JP0_0 to JP0_2, P1_8 to P1_11, P2_0, P2_1) | | | –30 | mA |
| | | | Per side8 (Total of JP0_6, P0_7 to P0_10, P1_4 to P1_7, P1_14, P1_15, P2_2 to P2_5, P8_0, P8_1, P8_3 to P8_9) | | | –30 | mA |
| | | | Total (EVCC: side1 to side4.) | | | –60 | mA |
| | | | Total (EVCC: side4 to side8.) | | | –60 | mA |
| | | PgA0 | Total (A0VSS) | | | –16 | mA |
| | | PgA1 | Total (A1VSS) | | | –16 | mA |
| Low-level output current | IOL | PgE | Per side1 (Total of P9_0 to P9_6, P20_0 to P20_5) | | | 30 | mA |
| | | | Per side2 (Total of P18_0 to P18_7) | | | 30 | mA |
| | | | Per side3 (Total of P10_6 to P10_14, P11_1, P11_2) | | | 30 | mA |
| | | | Per side4 (Total of P11_3 to P11_7, P11_15, P12_0 to P12_2) | | | 30 | mA |
| | | | Per side5 (Total of P10_0 to P10_5) | | | 30 | mA |
| | | | Per side6 (Total of P10_15, P11_0, P11_8 to P11_14, P12_3 to P12_5) | | | 30 | mA |
| | | | Per side7 (Total of P0_0 to P0_6, P0_11 to P0_14, P1_0 to P1_3, P1_12, P1_13, P2_6) | | | 30 | mA |
| | | | Per side8 (Total of JP0_0 to JP0_5, P1_8 to P1_11, P2_0, P2_1, P8_2, P8_10 to P8_12) | | | 30 | mA |
| | | | Per side9 (Total of JP0_6, P0_7 to P0_10, P2_2, P2_3) | | | 30 | mA |
| | | | Per side10 (Total of P1_4 to P1_7, P1_14, P1_15, P2_4, P2_5, P8_0, P8_1, P8_3 to P8_9) | | | 30 | mA |
| | | | Total (EVCC: side1 to side5.) | | | 60 | mA |
| | | | Total (EVCC: side5 to side10.) | | | 60 | mA |
| | | PgA0 | Total (A0VSS) | | | 16 | mA |
| | | PgA1 | Total (A1VSS) | | | 16 | mA |

**Note:** For detail of the definition of “side” and “total”, see **Section 40.2.3, Port Current**.

### 40.9.1.2 144 pin

| Item | Symbol | Pin Group | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| High-level output current | IOH | PgE | Per side1 (Total of P9_0 to P9_6, P20_4, P20_5) | | | –17 | mA |
| | | | Per side2 (Total of P10_6 to P10_9, P18_0 to P18_3) | | | –30 | mA |
| | | | Per side3 (Total of P10_10 to P10_14, P11_1 to P11_7, P11_15, P12_0 to P12_2) | | | –30 | mA |
| | | | Per side4 (Total of P0_0 to P0_3, P10_0 to P10_5, P10_15, P11_0, P11_8 to P11_14) | | | –30 | mA |
| | | | Per side5 (Total of JP0_3 to JP0_5, P0_4 to P0_6, P0_11 to P0_14, P1_0 to P1_3, P8_2, P8_10 to P8_12) | | | –30 | mA |
| | | | Per side6 (Total of JP0_0 to JP0_2, P1_8 to P1_11) | | | –30 | mA |
| | | | Per side7 (Total of JP0_6, P0_7 to P0_10, P1_4 to P1_7, P8_0, P8_1, P8_3 to P8_9) | | | –30 | mA |
| | | | Total (EVCC: side1 to side4.) | | | –60 | mA |
| | | | Total (EVCC: side4 to side7.) | | | –60 | mA |
| | | PgA0 | Total (A0VREF) | | | –16 | mA |
| | | PgA1 | Total (A1VREF) | | | –16 | mA |
| Low-level output current | IOL | PgE | Per side1 (Total of P9_0 to P9_6, P20_4, P20_5) | | | 17 | mA |
| | | | Per side2 (Total of P18_0 to P18_3) | | | 20 | mA |
| | | | Per side3 (Total of P10_6 to P10_14, P11_1, P11_2) | | | 30 | mA |
| | | | Per side4 (Total of P11_3 to P11_7, P11_15, P12_0 to P12_2) | | | 30 | mA |
| | | | Per side5 (Total of P0_0 to P0_6, P0_11 to P0_14, P1_0 to P1_3, P10_0 to P10_5, P10_15, P11_0, P11_8 to P11_14) | | | 30 | mA |
| | | | Per side6 (Total of JP0_0 to JP0_5, P1_8 to P1_11, P8_2, P8_10 to P8_12) | | | 30 | mA |
| | | | Per side7 (Total of JP0_6, P0_7 to P0_10) | | | 25 | mA |
| | | | Per side8 (Total of P1_4 to P1_7, P8_0, P8_1, P8_3 to P8_9) | | | 29 | mA |
| | | | Total (EVCC: side1 to side5.) | | | 60 | mA |
| | | | Total (EVCC: side5 to side8.) | | | 60 | mA |
| | | PgA0 | Total (A0VSS) | | | 16 | mA |
| | | PgA1 | Total (A1VSS) | | | 16 | mA |

**Note:** For detail of the definition of “side” and “total”, see **Section 40.2.3, Port Current**.

### 40.9.1.3 100 pin

| Item | Symbol | Pin Group | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| High-level output current | IOH | PgE | Per side (Total of P9_0 to P9_6) | | | –7 | mA |
| | | | Per side (Total of P0_0 to P0_3, P10_3 to P10_5, P10_15, P11_0) | | | –30 | mA |
| | | | Per side (Total of JP0_3 to JP0_5, P0_4 to P0_6, P0_11 to P0_14, P8_2, P8_10 to P8_12) | | | –30 | mA |
| | | | Per side (Total of JP0_0 to JP0_2) | | | –11 | mA |
| | | | Per side (Total of P0_7 to P0_10, P8_0, P8_1, P8_3 to P8_9) | | | –29 | mA |
| | | | Per side (Total of P10_6 to P10_9) | | | –20 | mA |
| | | | Per side (Total of P10_10 to P10_14, P11_1 to P11_7) | | | –30 | mA |
| | | | Per side (Total of P10_0 to P10_2) | | | –15 | mA |
| | | | Total (EVCC) | | | –60 | mA |
| | | PgA0 | Total (A0VREF) | | | –16 | mA |
| Low-level output current | IOL | PgE | Per side (Total of P9_0 to P9_6) | | | 7 | mA |
| | | | Per side (Total of P0_0 to P0_6, P0_11 to P0_14, P10_3 to P10_5, P10_15, P11_0) | | | 30 | mA |
| | | | Per side (Total of JP0_0 to JP0_5, P8_2, P8_10 to P8_12) | | | 26 | mA |
| | | | Per side (Total of P0_7 to P0_10) | | | 20 | mA |
| | | | Per side (Total of P8_0, P8_1, P8_3 to P8_9 | | | 9 | mA |
| | | | Per side (Total of P10_6 to P10_14, P11_1, P11_2) | | | 30 | mA |
| | | | Per side (Total of P11_3 to P11_7) | | | 25 | mA |
| | | | Per side (Total of P10_0 to P10_2) | | | 15 | mA |
| | | | Total (EVSS) | | | 60 | mA |
| | | PgA0 | Total (A0VSS) | | | 16 | mA |

**Note:** For detail of the definition of “side” and “total”, see **Section 40.2.3, Port Current**.

## 40.10 Power Supply Currents

**Condition:** REGVCC, EVCC, A0VREF and A1VREF total current. But the I/O buffer is stopped.

### ECO

| Item | Symbol | Condition | | | | MIN. | TYP.*1 | MAX. | Unit |
|---|---|---|---|---|---|---|---|---|---|
| | | CPU | PLL | Tj | Peripheral*2 | | | | |
| RUN mode current | IDDR | Run (80 MHz) | Run | –40 to 150°C | Run(#1) | | 26 | 74 | mA |
| | | | | 25°C | Stop(#1) | | 20 | | mA |
| RUN mode current (During data/code flash programming) | IDDR3 | Run (80 MHz) | Run | –40 to 150°C | Run(#2) | | 37 | 89 | mA |
| HALT mode current | IDDH | Run (80 MHz) | Run | –40 to 150°C | Run(#3) | | 23 | 72 | mA |

### ADVANCED, PREMIUM

| Item | Symbol | Condition | | | | MIN. | TYP.*1 | MAX. | Unit |
|---|---|---|---|---|---|---|---|---|---|
| | | CPU | PLL | Tj | Peripheral*2 | | | | |
| RUN mode current | IDDR | Run (120 MHz) | Run | –40 to 150°C | Run(#1) | | 32 | 81 | mA |
| | | | | 25°C | Stop(#1) | | 26 | | mA |
| RUN mode current (During data/code flash programming) | IDDR3 | Run (120 MHz) | Run | –40 to 150°C | Run(#2) | | 43 | 96 | mA |
| HALT mode current | IDDH | Run (120 MHz) | Run | –40 to 150°C | Run(#3) | | 29 | 77 | mA |

| Item | Symbol | Condition | | | | MIN. | TYP.*1 | MAX. | Unit |
|---|---|---|---|---|---|---|---|---|---|
| | | CPU | PLL | Tj | Peripheral*2 | | | | |
| STOP mode current | IDDS | Stop | Stop | –40 to 90°C | Stop(#2) | | 0.7 | 12 | mA |
| | | | | 110°C | Stop(#2) | | | 17 | mA |
| | | | | 135°C | Stop(#2) | | | 31 | mA |
| DeepSTOP mode current | IDDDS | Power off | Power off | –40 to 85°C | Stop(#3) | | 50 | 470 | µA |
| | | | | 105°C | Stop(#3) | | | 830 | µA |
| | | | | 125°C | Stop(#3) | | | 1370 | µA |
| Cyclic RUN mode current | IDDCR | Run (HS IntOSC) | Stop | –40 to 90°C | Run(#4) | | 3.6 | 21 | mA |
| | | | | 115°C | Run(#4) | | | 28 | mA |
| | | | | 135°C | Run(#4) | | | 40 | mA |
| Cyclic STOP mode current | IDDCS | Stop | Stop | –40 to 90°C | Run(#5) | | 1.1 | 13 | mA |
| | | | | 110°C | Run(#5) | | | 18 | mA |
| | | | | 135°C | Run(#5) | | | 32 | mA |

Note 1. The condition of “TYP.” shows the specification with the following conditions. Also, the value is just for reference only.
- Tj = 25°C
- REGVCC = EVCC = A0VREF = A1VREF = 5.0 V
- AWOVSS = EVSS = A0VSS = A1VSS = 0 V

Note 2. Operating condition of each peripheral function is shown in the table of next page.

**Caution:** It must be ensured that the junction temperature in the Ta range remains below Tj≤150°C and does not exceed its limit under application conditions (thermal resistance, power supply current, peripheral current (if not included in power supply current), port output current and injection current).

| Function | | Run | | | | | Stop | | |
|---|---|---|---|---|---|---|---|---|---|
| | | (#1) | (#2) | (#3) | (#4) | (#5) | (#1) | (#2) | (#3) |
| AWO | MainOSC | Run | Run | Run | Stop | Stop | Run | Stop | Stop |
| | SubOSC | Stop | Stop | Stop | Stop | Stop | Stop | Stop | Stop |
| | HS IntOSC | Run | Run | Run | Run | Stop | Run | Stop | Stop |
| | FOUT | Stop | Stop | Stop | Stop | Stop | Stop | Stop | Stop |
| | LPS | Stop | Stop | Stop | Stop | Stop | Stop | Stop | Stop |
| | RRAM | Read/Write | Read/Write | No access | Fetch | No access | Read/Write | No access | No access |
| | WDTA0 | Stop | Stop | Stop | Stop | Stop | Stop | Stop | Stop |
| | TAUJ0 | Run | Run | Run | Run (LS IntOSC) | Run (LS IntOSC) | Stop | Stop | Stop |
| | RTCA0 | Run | Run | Run | Run (LS IntOSC) | Run (LS IntOSC) | Stop | Stop | Stop |
| | CLMA0 | Run | Run | Run | Run | Stop | Stop | Stop | Stop |
| | CLMA1 | Run | Run | Run | Stop | Stop | Stop | Stop | Stop |
| | ADCA0 | Run*1 | Run*1 | Run*1 | Stop | Stop | Stop | Stop | Stop |
| ISO | CPU | Run (PLL) | Run (PLL) | Halt (PLL) | Run (HS IntOSC) | Stop | Run (PLL) | Stop | Power off |
| | DMA | Run | Run | Run | Stop | Stop | Stop | Stop | |
| | PLL | Run | Run | Run | Stop | Stop | Run | Stop | |
| | Code flash | Fetch | Fetch | No access | No access | No access | Fetch | No access | |
| | Data flash | Read | Write\Erase | No access | No access | No access | Read | No access | |
| | LRAM | Read/Write | Read/Write | No access | No access | No access | Read/Write | No access | |
| | OSTM0 | Run | Run | Run | Stop | Stop | Stop | Stop | |
| | WDTA1 | Stop | Stop | Stop | Stop | Stop | Stop | Stop | |
| | TAUD0 | Run | Run | Run | Stop | Stop | Stop | Stop | |
| | TAUBn | Run | Run | Run | Stop | Stop | Stop | Stop | |
| | TAUJ1 | Run | Run | Run | Stop | Stop | Stop | Stop | |
| | TAPA, PIC | Stop | Stop | Stop | Stop | Stop | Stop | Stop | |
| | ENCA0 | Run | Run | Run | Stop | Stop | Stop | Stop | |
| | PWM-diag | Run | Run | Run | Stop | Stop | Stop | Stop | |
| | RLIN3n | Run | Run | Run | Stop | Stop | Stop | Stop | |
| | RLIN2n | Wait | Wait | Wait | Stop | Stop | Stop | Stop | |
| | RS-CANn | Wait | Wait | Wait | Stop | Stop | Stop | Stop | |
| | CSIGn | Run | Run | Run | Stop | Stop | Stop | Stop | |
| | CSIHn | Run | Run | Run | Stop | Stop | Stop | Stop | |
| | RIIC0 | Wait | Wait | Wait | Stop | Stop | Stop | Stop | |
| | CLMA2 | Run | Run | Run | Stop | Stop | Stop | Stop | |
| | ADCA1 | Run | Run | Run | Stop | Stop | Stop | Stop | |

Note 1. T&H used.

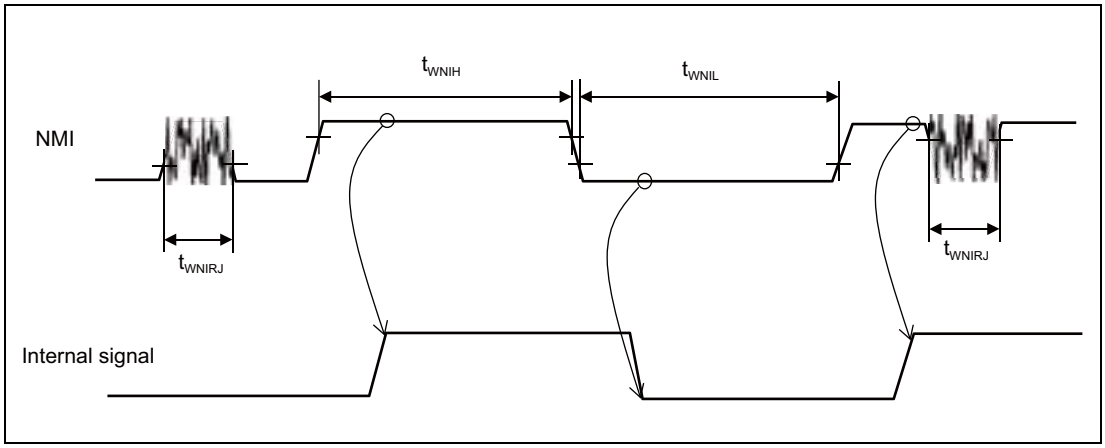
## 40.11 Interrupt Timing

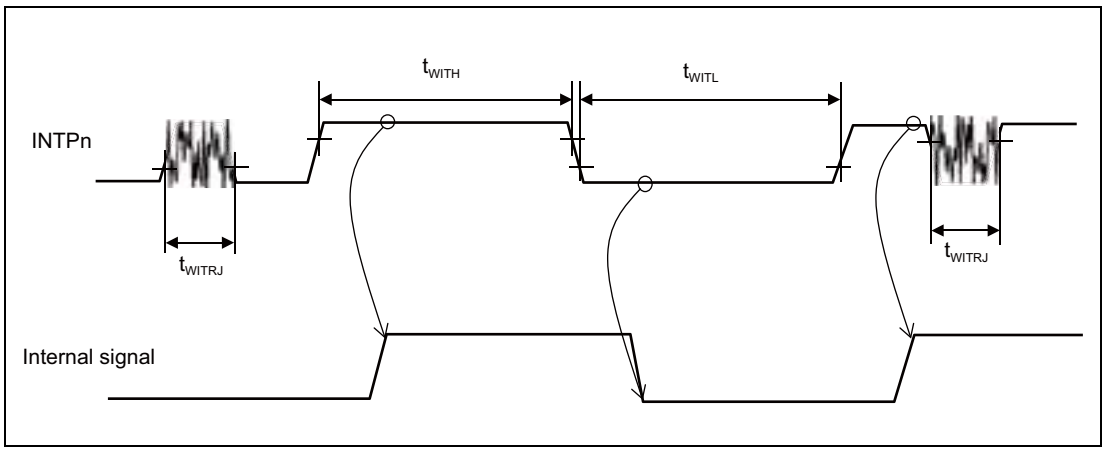
**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| NMI input high/low level width*1 | $t_{WNIH}$/ $t_{WNIL}$ | Edge detection mode | 600 | | | ns |
| | | Level detection mode (EMCLK is operated by HS IntOSC) | 756 | | | ns |
| | | Level detection mode (EMCLK is operated by LS IntOSC) | 5.13 | | | µs |
| NMI pulse rejection*2 | $t_{WNIRJ}$ | | 100 | | | ns |
| INTPn input high/low level width*1 | $t_{WITH}$/ $t_{WITL}$ | Edge detection mode | 600 | | | ns |
| | | Level detection mode (EMCLK is operated by HS IntOSC) | 756 | | | ns |
| | | Level detection mode (EMCLK is operated by LS IntOSC) | 5.13 | | | µs |
| INTPn pulse rejection*2 | $t_{WITRJ}$ | | 100 | | | ns |

Note 1. NMI and INTPn input width is needed to ensure that the internal interrupt signal is activated.

Note 2. Pulses shorter than this minimum is ignored. This is reference value. Noise such as the figure can be filtered.



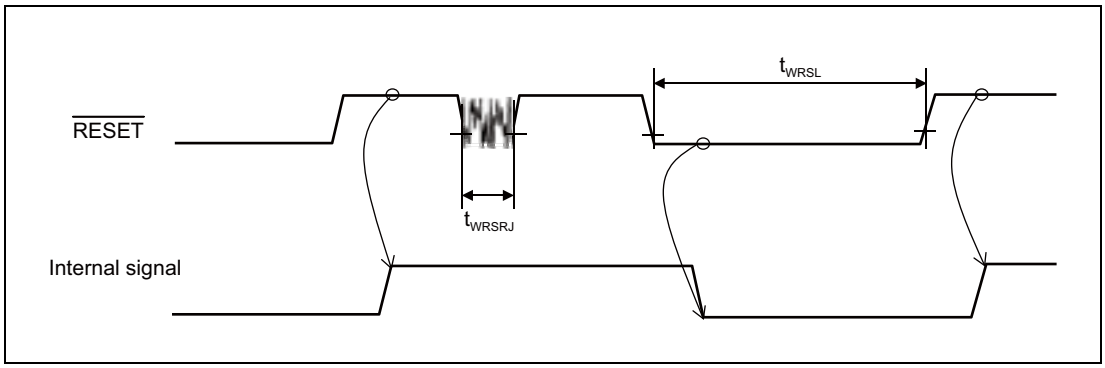
## 40.12 $\overline{\text{RESET}}$ Timing

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{RESET}}$ input low level width*1 | $t_{WRSL}$ | Except power on | 600 | | | ns |
| $\overline{\text{RESET}}$ pulse rejection*2 | $t_{WRSRJ}$ | | 100 | | | ns |

Note 1. $\overline{\text{RESET}}$ input width is needed to ensure that the internal reset signal is activated.

Note 2. Pulses shorter than this minimum is ignored. This is reference value. Noise such as the figure can be filtered.

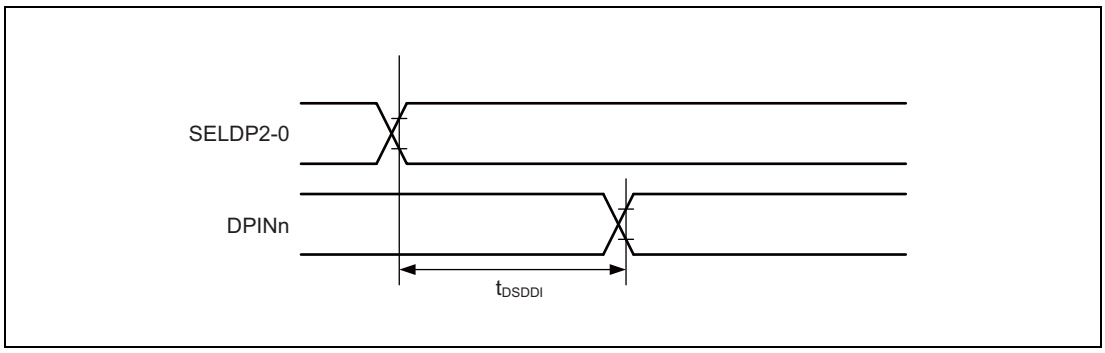



## 40.13 Low Power Sampler (DPIN input) Timing

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| DPINn input delay time (vs SELDP2-0) | $t_{DSDDI}$ | | | | 150 | ns |

n = 7 to 0



## 40.14 CSCXFOUT Timing

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF +/–30%, CISOVCL: 0.1µF +/–30%, Ta = –40 to (depend on the product) °C,
CL = 30 pF

<Output driver strength>

CSCXFOUT: Slow or fast mode (refer to the condition in the following table)

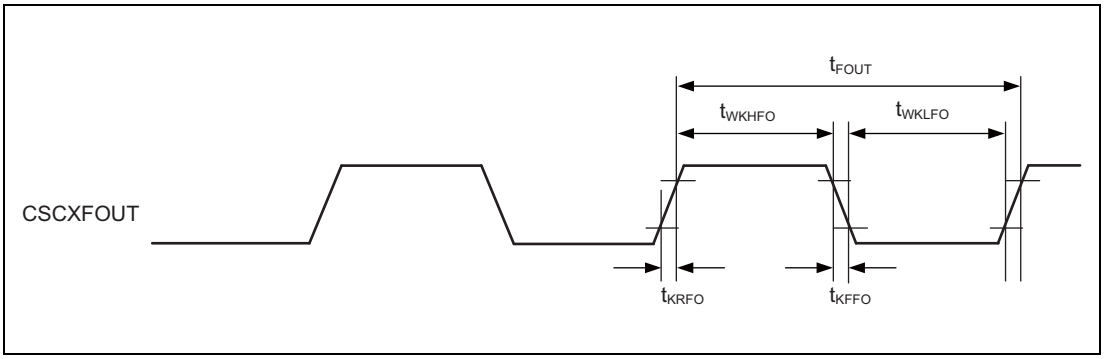
| Item | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| CSCXFOUT output cycle | $t_{FOUT}$ | Slow mode | | 100 (max. 10 MHz) | | | ns |
| | | Fast mode (Except JP0_3 pin)*1 | | 41.6 (max. 24 MHz) | | | ns |
| CSCXFOUT high level width | $t_{WKHFO}$ | Slow mode | N: 1*2 or even value*3 | $t_{FOUT}$ / 2 – 37 | | | ns |
| | | | N: Odd value (N ≥ 5)*3, *4 | $t_{FOUT}$ × (N+1) / 2N – 37 | | | ns |
| | | Fast mode (Except JP0_3 pin)*1 | N: 1*2 or even value*3 | $t_{FOUT}$ / 2 – 10 | | | ns |
| | | | N: Odd value (N ≥ 3)*3 | $t_{FOUT}$ × (N+1) / 2N – 10 | | | ns |
| CSCXFOUT low level width | $t_{WKLFO}$ | Slow mode | N: 1*2 or even value*3 | $t_{FOUT}$ / 2 – 37 | | | ns |
| | | | N: Odd value (N ≥ 5)*3, *4 | $t_{FOUT}$ × (N–1) / 2N – 37 | | | ns |
| | | Fast mode (Except JP0_3 pin)*1 | N: 1*2 or even value*3 | $t_{FOUT}$ / 2 – 10 | | | ns |
| | | | N: Odd value (N ≥ 3)*3 | $t_{FOUT}$ × (N–1) / 2N – 10 | | | ns |
| CSCXFOUT rise/ fall time | $t_{KRFO}$/ $t_{KFFO}$ | Slow mode | | | | 37 | ns |
| | | Fast mode (Except JP0_3 pin)*1 | | | | 10 | ns |

Note 1. JP0_3 does not support fast mode.

Note 2. When MainOSC, HS IntOSC, LS IntOSC or SubOSC is selected as source clock with the condition of N=1, the characteristics of output signal depends on the selected source clock. It is recommended to use output signal after evaluation on an actual environment.

Note 3. “N” is the value of “Clock divisor N” defined by FOUTDIV register.

Note 4. The selection of N = 3 is prohibited when slow mode is used.



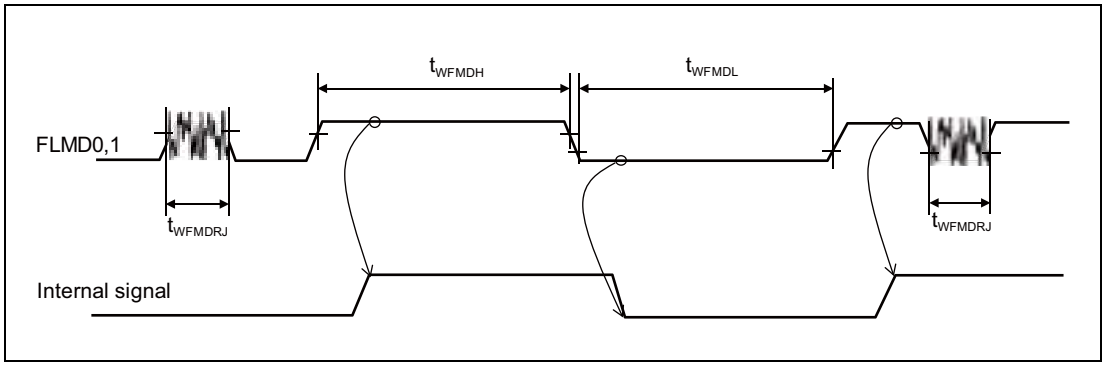
## 40.15 Mode Timing

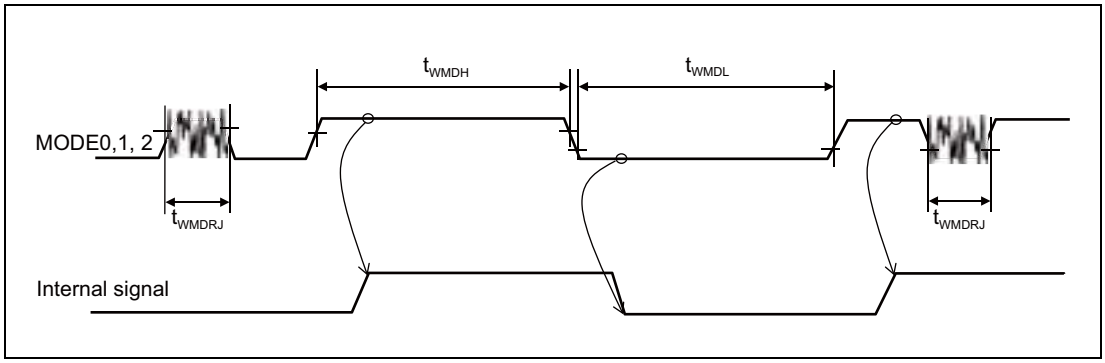
**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| FLMD0,1 input high/low level width*1 | $t_{WFMDH}$/ $t_{WFMDL}$ | | 600 | | | ns |
| FLMD0, 1 pulse rejection*2 | $t_{WFMDRJ}$ | | 100 | | | ns |
| MODE0, 1, 2 input high/low level width*1 | $t_{WMDH}$/ $t_{WMDL}$ | | 600 | | | ns |
| MODE0, 1, 2 pulse rejection*2 | $t_{WMDRJ}$ | | 100 | | | ns |

Note 1. FLMD0,1 and MODE0,1, 2 input width is needed to ensure that the internal mode signal is activated.

Note 2. Pulses shorter than this minimum is ignored. This is reference value. Noise such as the figure can be filtered.



## 40.16 Timer Timing

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

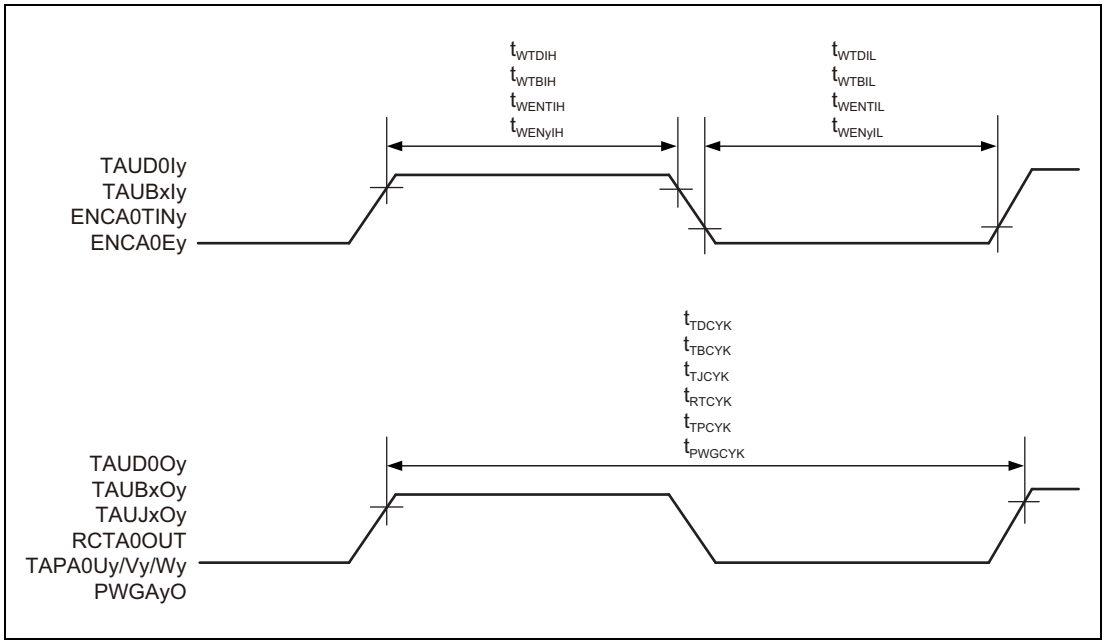
| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| TAUD0Iy input high/low level width (y = 0 to 15) | $t_{WTDIH}$/ $t_{WTDIL}$ | | n × Tsamp + 20[*1, *2] | | | ns |
| TAUD0Oy output cycle (y = 0 to 15) | $t_{TDCYK}$ | Slow mode | | | 10 | MHz |
| TAUBxIy input high/low level width (x = 0, 1, y = 0 to 15) | $t_{WTBIH}$/ $t_{WTBIL}$ | | n × Tsamp + 20[*1, *2] | | | ns |
| TAUBxOy output cycle (x = 0, 1, y = 0 to 15) | $t_{TBCYK}$ | Slow mode | | | 10 | MHz |
| TAUJxIy input high/low level width[*3] (x = 0, 1, y = 0 to 3) | $t_{WTJIH}$/ $t_{WTJIL}$ | | 600 | | | ns |
| TAUJxIy pulse rejection[*4] | $t_{WTIJRJ}$ | | 100 | | | ns |
| TAUJxOy output cycle (x = 0, 1, y = 0 to 3) | $t_{TJCYK}$ | Slow mode | | | 10 | MHz |
| RTCA0OUT output cycle | $t_{RTCYK}$ | | | 1 | | Hz |
| TAPA0ESO input high/low level width[*3] | $t_{WESIH}$/ $t_{WESIL}$ | | 600 | | | ns |
| TAPA0ESO pulse rejection[*4] | $t_{WESIRJ}$ | | 100 | | | ns |
| TAPA0Uy/Vy/Wy output cycle (y = P, N) | $t_{TPCYK}$ | Slow mode | | | 10 | MHz |
| ENCA0TINy input high/low level width (y = 0, 1) | $t_{WENTIH}$/ $t_{WENTIL}$ | | n × Tsamp + 20[*1] | | | ns |
| ENCA0Ey input high/low level width (y = 0,1,C) | $t_{WENyIH}$/ $t_{WENyIL}$ | | n × Tsamp + 20[*1] | | | ns |
| PWGAyO output cycle (y = 0 to 71) | $t_{PWGCYK}$ | Slow mode | | | 10 | MHz |

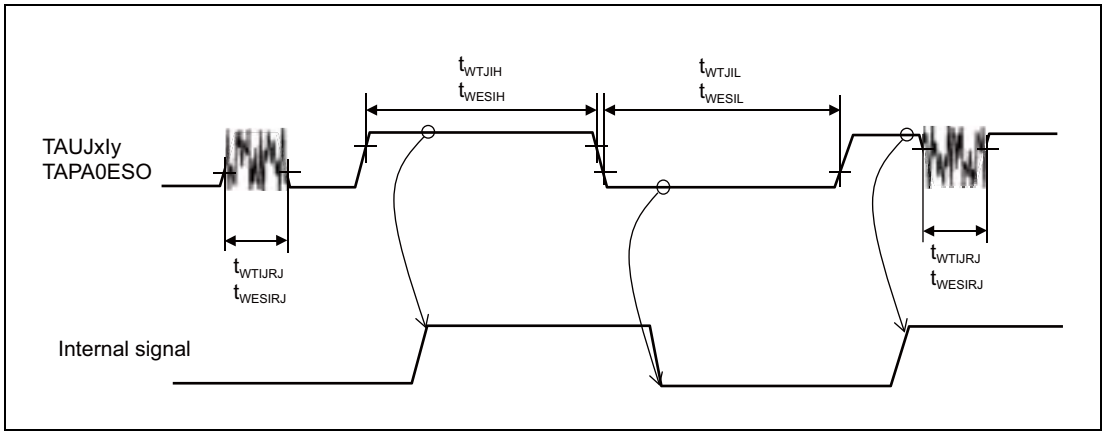
Note 1. n: Sampling number of the digital noise filter for each input.
Tsamp: Sampling time of the digital noise filter for each input.

Note 2. Input more than 1 count clock width of each timer counter channel.

Note 3. TAUJxIy and TAPA0ESO input width is needed to ensure that the internal timer input signal is activated.

Note 4. Pulses shorter than this minimum is ignored. This is reference value. Noise such as the figure can be filtered.


tWTDIH
tWTBIH
tWENTIH
tWENyIH
tWTDIL
tWTBIL
tWENTIL
tWENyIL
TAUD0Iy
TAUBxIy
ENCA0TINy
ENCA0Ey
tTDCYK
tTBCYK
tTJCYK
tRTCYK
tTPCYK
tPWGCYK
TAUD0Oy
TAUBxOy
TAUJxOy
RCTA0OUT
TAPA0Uy/Vy/Wy
PWGAyO


tWTJIH
tWESIH
tWTJIL
tWESIL
TAUJxIy
TAPA0ESO
tWTIJRJ
tWESIRJ
tWTIJRJ
tWESIRJ
Internal signal

## 40.17 RLIN2/RLIN3 timing

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| RLIN3 transfer rate | | LIN specification | 1 | | 20 | kbps |
| | | LIN extended baud rate | 1 | | 115.2*1 | kbps |
| | | UART function | | | 1.5 | Mbps |
| RLIN2 transfer rate | | LIN specification | 1 | | 20 | kbps |

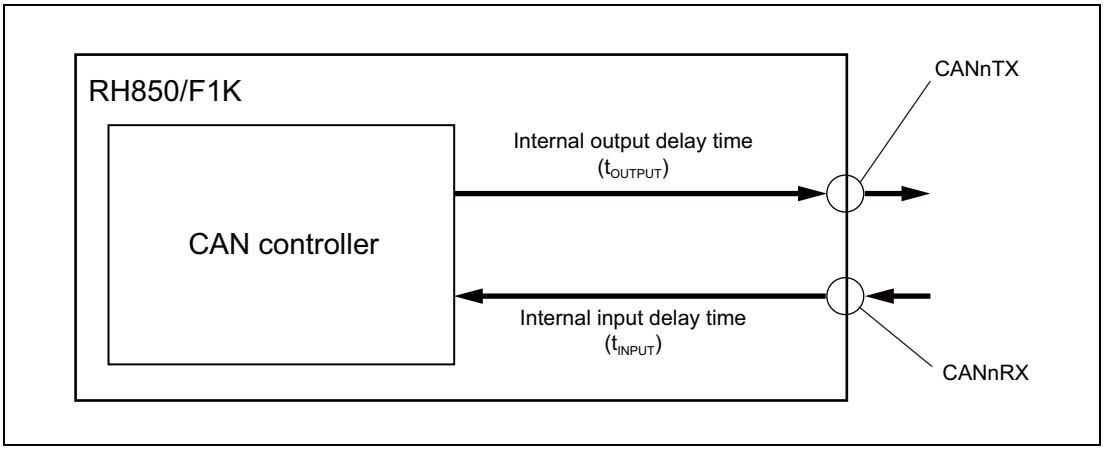
Note 1. The LIN extended baud rate is not part of the LIN standard specification.

## 40.18 RS-CAN Timing

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Transfer rate | | ECO, ADVANCED | | | 1 | Mbps |
| | | PREMIUM<br>Data rate | | | 5<br>(@ TQ = 8) | Mbps |
| Internal delay time*1 | $t_{NODE}$ | | | | 100 | ns |

Note 1. $t_{NODE}$ = Internal input delay time ($t_{INPUT}$) + Internal output delay time ($t_{OUTPUT}$)



# 40.19 CSI Timing

## 40.19.1 CSIG Timing

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF +/–30%, CISOVCL: 0.1µF +/–30%, Ta = –40 to (depend on the product) °C,
CL = 30 pF

**Table 40.8 CSIG Timing (Master Mode)**

<Output driver strength>

CSIGnSO, CSIGnSC (output): Fast mode

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Macro operation clock cycle time | $t_{KCYGn}$ | | 12.5 (max. 80 MHz) | | | ns |
| CSIGnSC cycle time | $t_{KCYMGn}$ | | 100 | | | ns |
| CSIGnSC high level width | $t_{KWHMGn}$ | | 0.5 × $t_{KCYMGn}$ – 10 | | | ns |
| CSIGnSC low level width | $t_{KWLMGn}$ | | 0.5 × $t_{KCYMGn}$ – 10 | | | ns |
| CSIGnSI setup time (vs. CSIGnSC) | $t_{SSIMGn}$ | | 30 | | | ns |
| CSIGnSI hold time (vs. CSIGnSC) | $t_{HSIMGn}$ | | 0 | | | ns |
| CSIGnSO output delay (vs. CSIGnSC) | $t_{DSOMGn}$ | | | | 7 | ns |
| CSIGnRYI setup time (vs. CSIGnSC) | $t_{SRYIGn}$ | CSIGnCTL1.CSIGnSIT = x<br>CSIGnCTL1.CSIGnHSE = 1 | 2 × $t_{KCYGn}$ + 25 | | | ns |
| CSIGnRYI high level width | $t_{WRYIGn}$ | CSIGnCTL1.CSIGnHSE = 1 | $t_{KCYGn}$ + 5 | | | ns |

n = 0, 1

**Table 40.9 CSIG Timing (Slave Mode)**

<Output driver strength>

CSIGnSO: Fast mode

CSIGnRYO: Slow mode

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Macro operation clock cycle time | $t_{KCYGn}$ | | 12.5 (max. 80 MHz) | | | ns |
| CSIGnSC cycle time | $t_{KCYSGn}$ | | 200 | | | ns |
| CSIGnSC high level width | $t_{KWHSGn}$ | | 0.5 × $t_{KCYSGn}$ – 10 | | | ns |
| CSIGnSC low level width | $t_{KWLSGn}$ | | 0.5 × $t_{KCYSGn}$ – 10 | | | ns |
| CSIGnSI setup time (vs. CSIGnSC) | $t_{SSISGn}$ | | 20 | | | ns |
| CSIGnSI hold time (vs. CSIGnSC) | $t_{HSISGn}$ | | $t_{KCYGn}$ + 5 | | | ns |
| CSIGnSO output delay (vs. CSIGnSC) | $t_{DSOSGn}$ | | | | 30 | ns |
| CSIGnRYO output delay | $t_{SRYOGn}$ | | | | 38 | ns |
| $\overline{\text{CSIGnSSI}}$ setup time (vs.CSIGnSC) | $t_{SSSISGn}$ | | 0.5 × $t_{KCYSGn}$ – 5 | | | ns |
| $\overline{\text{CSIGnSSI}}$ hold time (vs. CSIGnSC) | $t_{HSSISGn}$ | | $t_{KCYGn}$ + 5 | | | ns |

n = 0, 1

## 40.19.2 CSIH Timing

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF +/–30%, CISOVCL: 0.1µF +/–30%, Ta = –40 to (depend on the product) °C,
CL = 30 pF

**Table 40.10 CSIH Timing (Master Mode)**

<Output driver strength>

CSIHnSO, CSIHnSC (output): Fast mode (CL = 100pF@n=0 / 50pF@n=1-3)

CSIHnCSSm: Slow mode

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Macro Operation clock cycle time | $t_{KCYHn}$ | | 12.5 (max. 80 MHz) | | | ns |
| CSIHnSC cycle time | $t_{KCYMHn}$ | | 100 | | | ns |
| CSIHnSC high level width | $t_{KWHMHn}$ | | 0.5 × $t_{KCYMHn}$ – 10 | | | ns |
| CSIHnSC low level width | $t_{KWLMHn}$ | | 0.5 × $t_{KCYMHn}$ – 10 | | | ns |
| CSIHnSI setup time (vs. CSIHnSC) | $t_{SSIMHn}$ | SI positive edge mode (CSIHnCTL1.CSIHnSLRS = 0) | 19 | | | ns |
| | | SI negative edge mode (CSIHnCTL1.CSIHnSLRS = 1) | 14 | | | ns |
| CSIHnSI hold time (vs. CSIHnSC) | $t_{HSIMHn}$ | SI positive edge mode (CSIHnCTL1.CSIHnSLRS = 0) | 0 | | | ns |
| | | SI negative edge mode (CSIHnCTL1.CSIHnSLRS = 1) | $t_{KCYHn}$/2 | | | ns |
| CSIHnSO output delay (vs. CSIHnSC) | $t_{DSOMHn}$ | | | | 7 | ns |
| CSIHnRYI setup time (vs. CSIHnSC) | $t_{SRYIHn}$ | CSIHnCTL1.CSIHnSIT = x CSIHnCTL1.CSIHnHSE = 1 | 2 × $t_{KCYHn}$ + 25 | | | ns |
| CSIHnRYI high level width | $t_{WRYIHn}$ | CSIHnCTL1.CSIHnHSE = 1 | $t_{KCYHn}$ + 5 | | | ns |
| CSIHnCSS0-7 inactive width | $t_{WSCSBHn}$ | | CSIDLE × $t_{KCYMHn}$ – 15 | | | ns |
| CSIHnCSS0-7 setup time (vs. CSIHnSC) | $t_{SSCSBHn0}$ | CSIHnCFGx.CSIHnDAP = 0 | CSSETUP × $t_{KCYMHn}$ – 23 | | | ns |
| | $t_{SSCSBHn1}$ | CSIHnCFGx.CSIHnDAP = 1 | (CSSETUP + 0.5) × $t_{KCYMHn}$ – 23 | | | ns |
| CSIHnCSS0-7 hold time (vs. CSIHnSC) | $t_{HSCSBHn0}$ | CSIHnCTL1.CSIHnSIT = 0 | CSSHOLD × $t_{KCYMHn}$ – 5 | | | ns |
| | $t_{HSCSBHn1}$ | CSIHnCTL1.CSIHnSIT = 1 | (CSSHOLD + 0.5) × $t_{KCYMHn}$ – 5 | | | ns |

n = 0 to 3

**NOTE**

CSIDLE: Setting value of CSIHnCFGx.CSIHnIDx[2:0]

CSSETUP: Setting value of CSIHnCFGx.CSIHnSPx[3:0]

CSSHOLD: Setting value of CSIHnCFGx.CSIHnHDx[3:0]

x: Depends on number of the chip select signals.

**CAUTION**

When the serial clock level is changed during the communication (CSIHnCFGx.CSIHnCKPx) and the IDLE has a setting of 0.5 transmission clock cycles, an inactive width time $t_{WSCSBHn}$ of "0.5 × $t_{KCYMHn}$" is added.

### Table 40.11 CSIH Timing (Slave Mode)

<Output driver strength>
CSIHnSO, CSIHnSC (output): Fast mode
CSIHnRYO: Slow mode

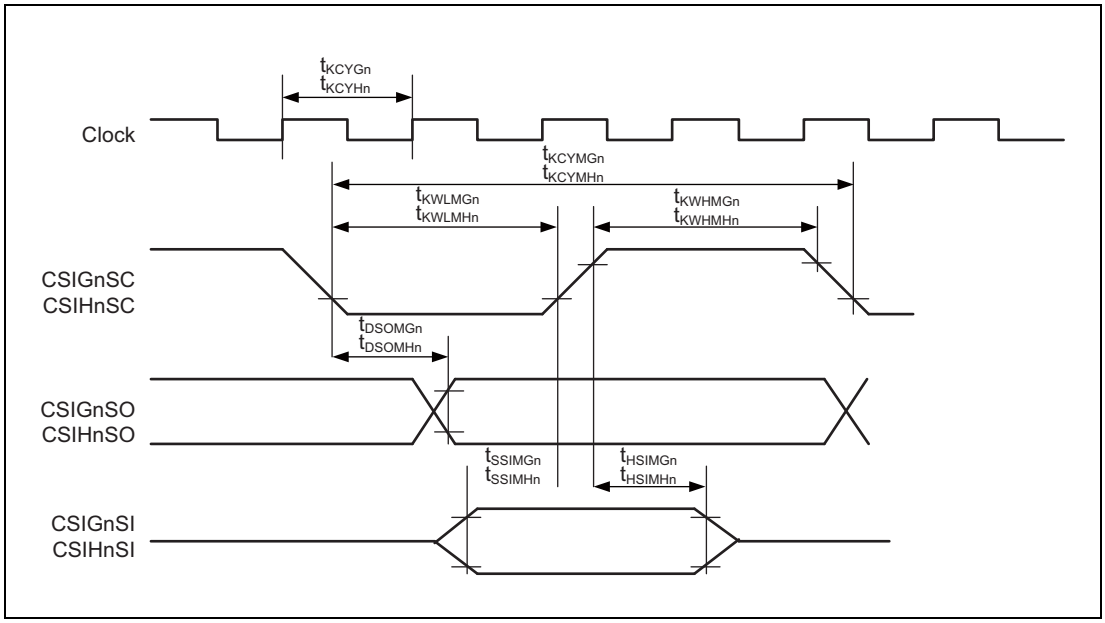
| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Macro Operation clock cycle time | $t_{KCYHn}$ | | 12.5 (max. 80 MHz) | | | ns |
| CSIHnSC cycle time | $t_{KCYSHn}$ | | 200 | | | ns |
| CSIHnSC high level width | $t_{KWHSHn}$ | | $0.5 \times t_{KCYSHn} - 10$ | | | ns |
| CSIHnSC low level width | $t_{KWLSHn}$ | | $0.5 \times t_{KCYSHn} - 10$ | | | ns |
| CSIHnSI setup time (vs. CSIHnSC) | $t_{SSISHn}$ | | 20 | | | ns |
| CSIHnSI hold time (vs. CSIHnSC) | $t_{HSISHn}$ | | $t_{KCYHn} + 5$ | | | ns |
| CSIHnSO output delay (vs. CSIHnSC) | $t_{DSOSHn}$ | | | | 30 | ns |
| CSIHnRYO output delay | $t_{SRYOHn}$ | | | | 38 | ns |
| $\overline{\text{CSIHnSSI}}$ setup time (vs. CSIHnSC) | $t_{SSSISHn}$ | | $0.5 \times t_{KCYSHn} - 5$ | | | ns |
| $\overline{\text{CSIHnSSI}}$ hold time (vs. CSIHnSC) | $t_{HSSISHn}$ | | $t_{KCYHn} + 5$ | | | ns |

n = 0 to 3

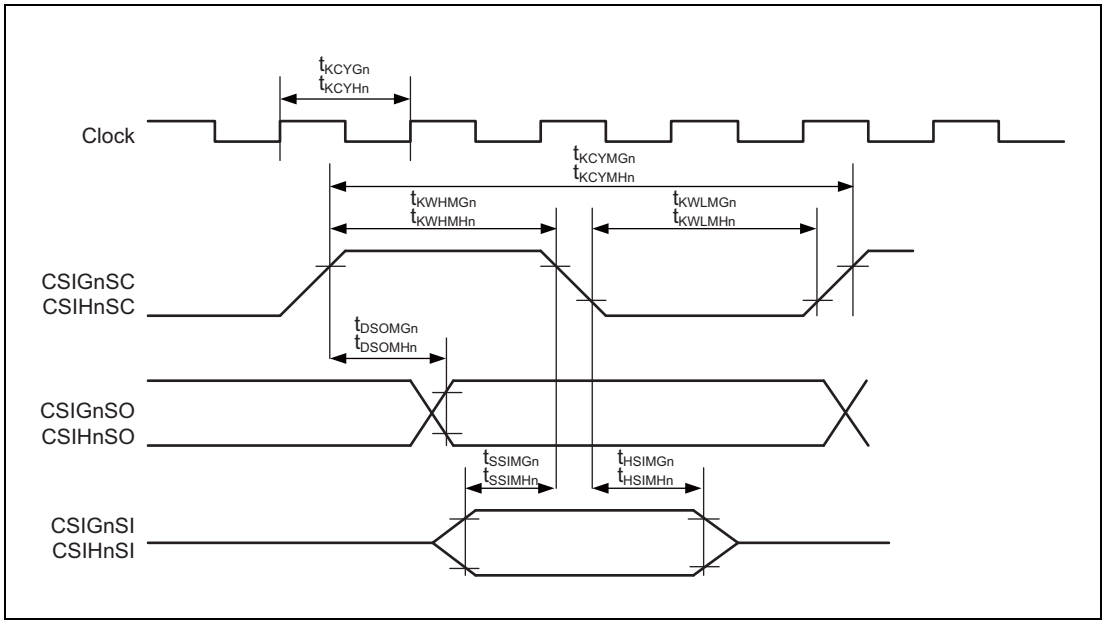
### (1) SCKO/SI/SO

**Master mode:**

- CSIG (CSIGnCTL1: CSIGnCKR/CSIGnCFG0: CSIGnDAP0 = 0/0 or 1/1)
- CSIH (CSIHnCFGm: CSIHnCKPm/CSIHnCFGm: CSIHnDAPm = 0/0 or 1/1)




- CSIG (CSIGnCTL1: CSIGnCKR/CSIGnCFG0: CSIGnDAP0 = 1/0 or 0/1)
- CSIH (CSIHnCFGm: CSIHnCKPm/CSIHnCFGm: CSIHnDAPm = 1/0 or 0/1)

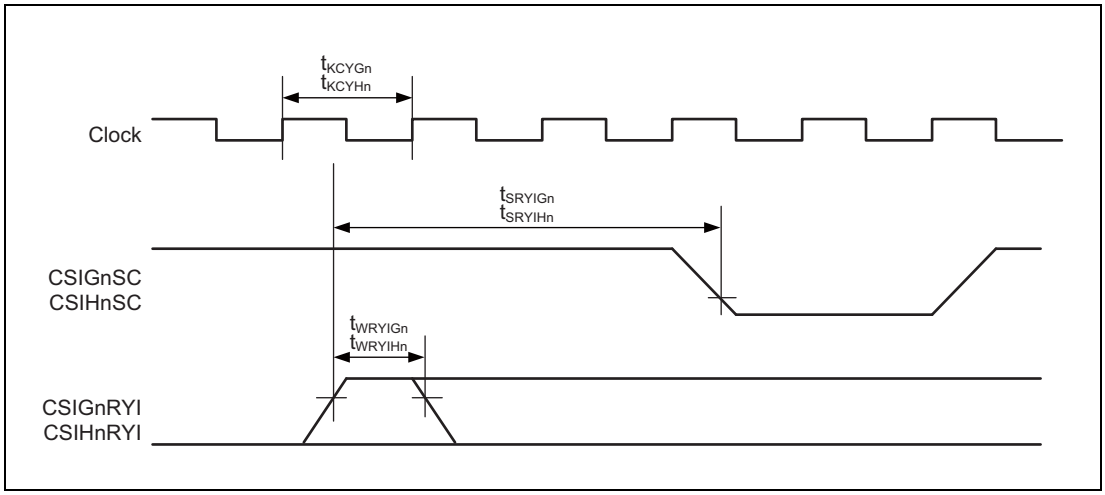


**(2) RYI**

- CSIG: Only master mode (CSIGnCTL1: CSIGnHSE = 1, CSIGnCTL1: CSIGnSIT = 0)
- CSIH: Only master mode (CSIHnCTL1: CSIHnHSE = 1, CSIHnCTL1: CSIHnSIT = 0)

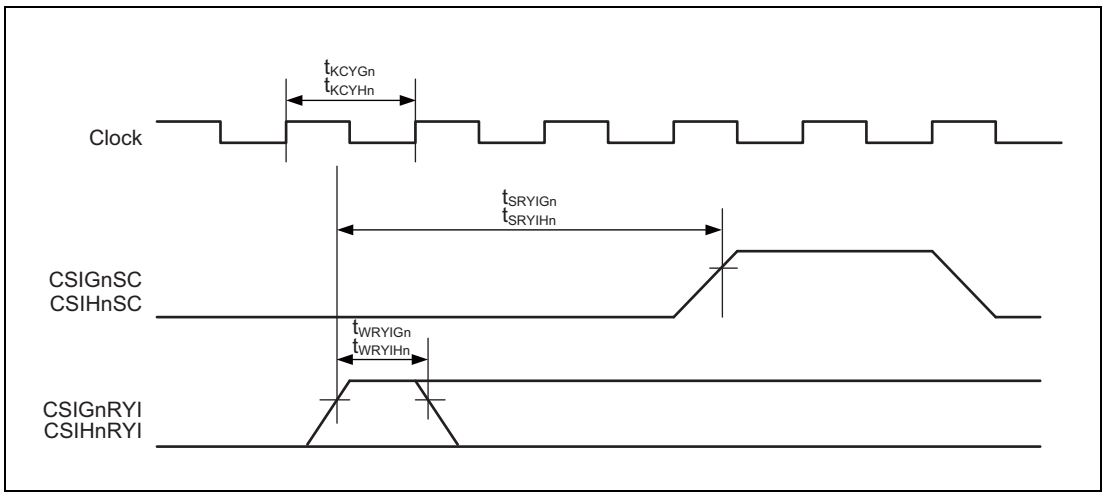
– CSIG (CSIGnCTL1: CSIGnCKR = 0)

– CSIH (CSIHnCFGm: CSIHnCKPm = 0)




– CSIG (CSIGnCTL1: CSIGnCKR = 1)

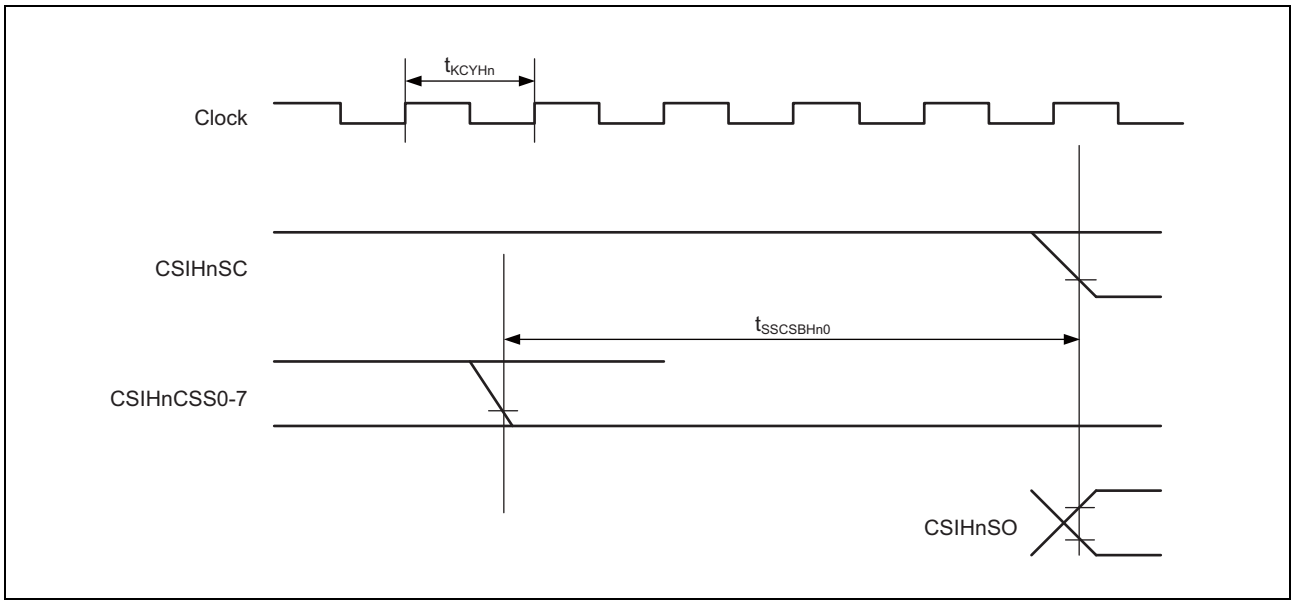
– CSIH (CSIHnCFGm: CSIHnCKPm = 1)



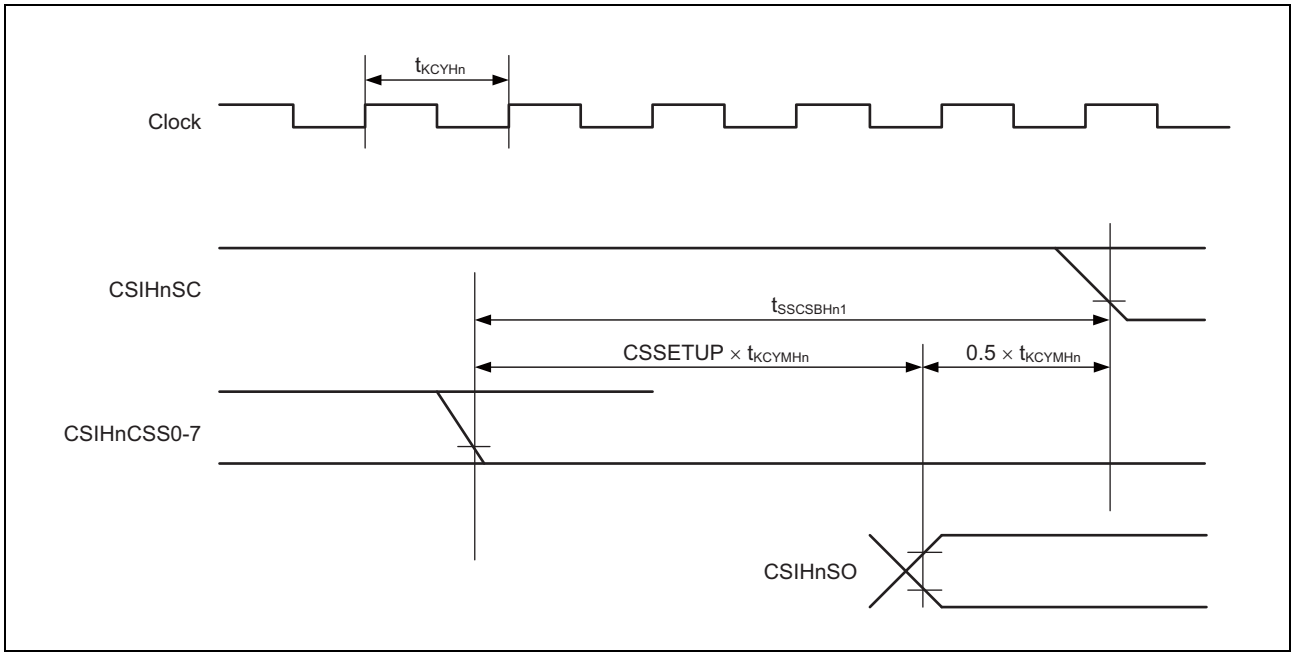
### (3) CSSn

**Only master mode (setup time):**

- CSIHnCFGm: CSIHnCKPm = 0, CSIHnCFGm: CSIHnDAPm = 0

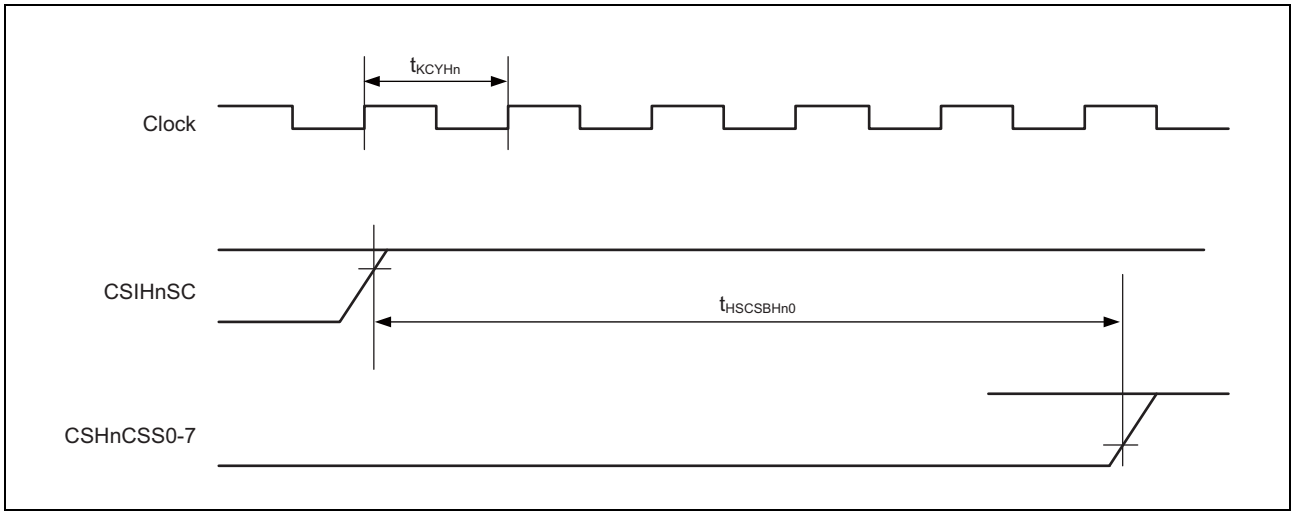



- CSIHnCFGm: CSIHnCKPm = 0, CSIHnCFGm: CSIHnDAPm = 1

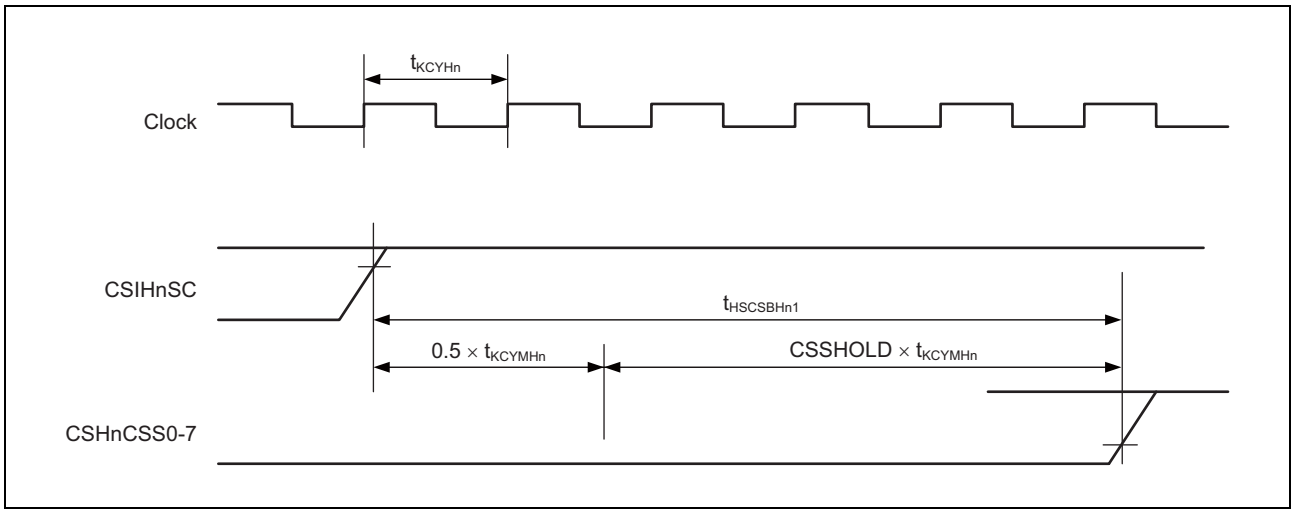


**Only master mode (hold time):**

- CSIHnCTL1: CSIHnSIT = 0, CSIHnCFGm: CSIHnCKPm = 0, CSIHnCFGm: CSIHnDAPm = 0



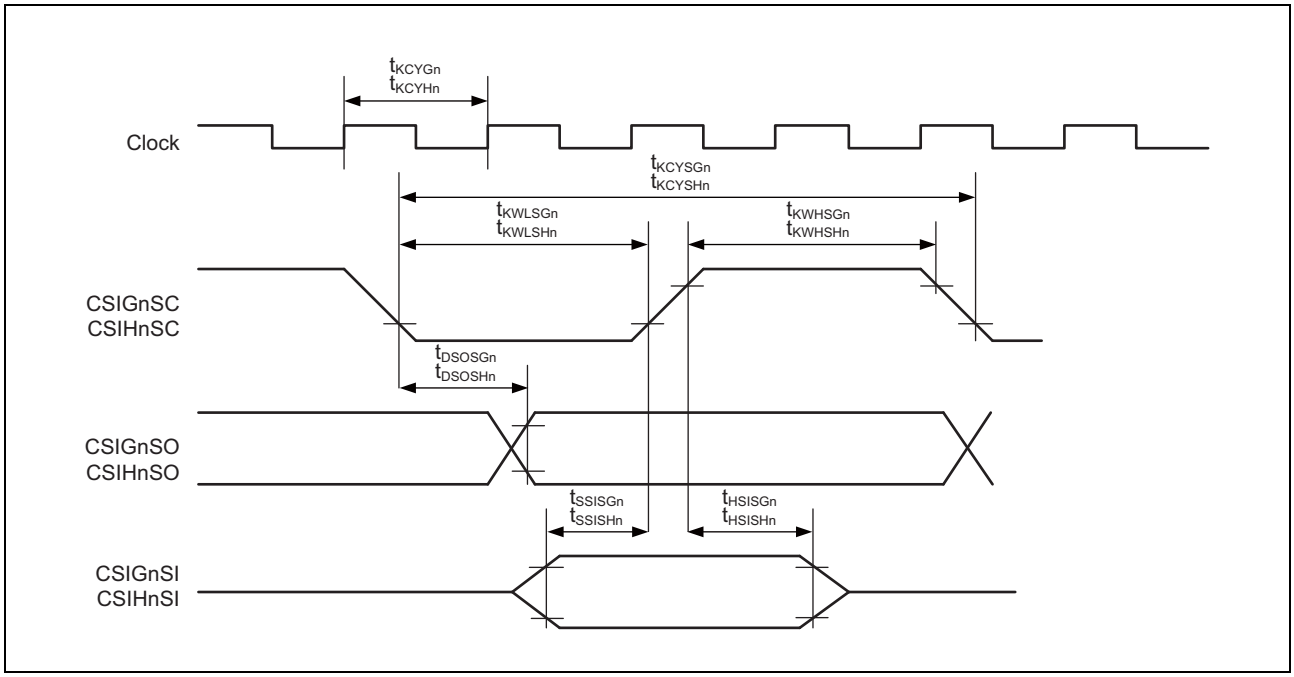

- CSIHnCTL1: CSIHnSIT = 1, CSIHnCFGm: CSIHnCKPm = 0, CSIHnCFGm: CSIHnDAPm = 0



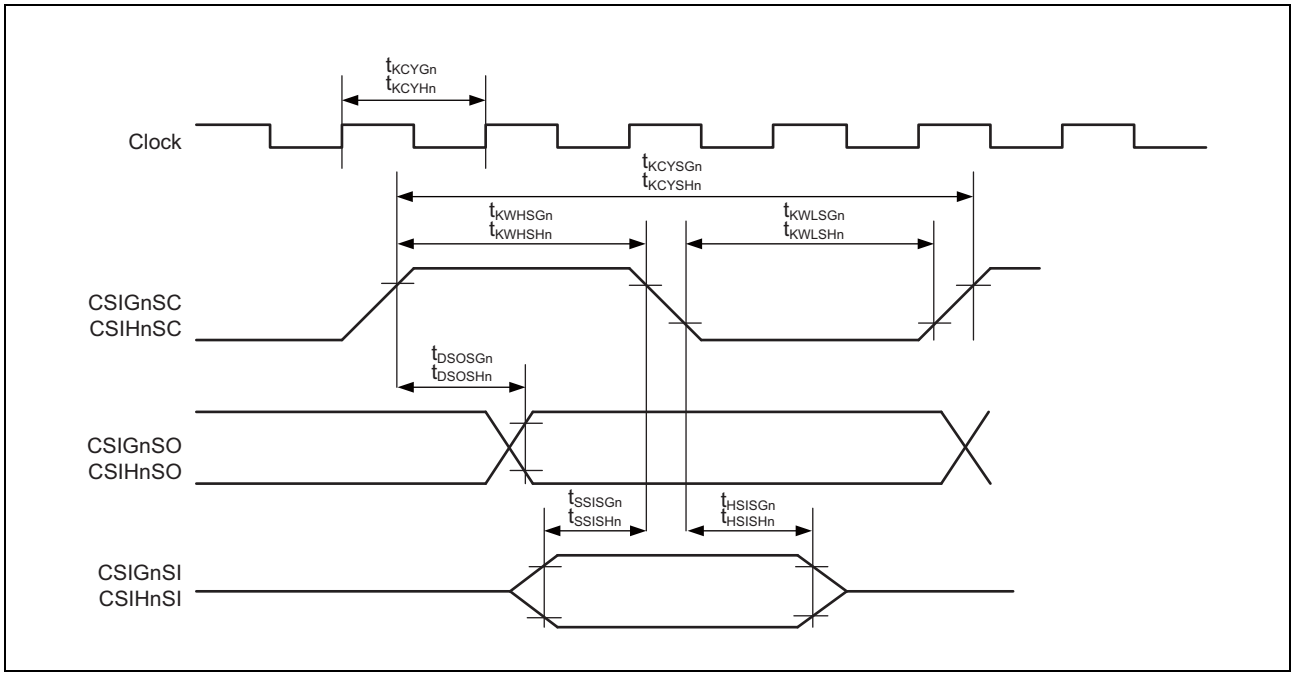
### (4) SCKO/SI/SO

**Slave mode:**

- CSIG (CSIGnCTL1: CSIGnCKR/CSIGnCFG0: CSIGnDAP0 = 0/0 or 1/1)
- CSIH (CSIHnCFGm: CSIHnCKPm/CSIHnCFGm: CSIHnDAPm = 0/0 or 1/1)

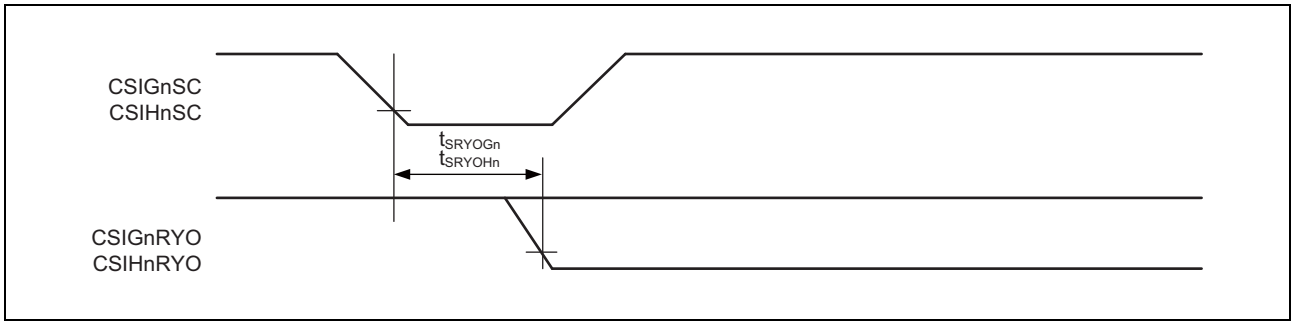



- CSIG (CSIGnCTL1: CSIGnCKR/CSIGnCFG0: CSIGnDAP0 = 1/0 or 0/1)
- CSIH (CSIHnCFGm: CSIHnCKPm/CSIHnCFGm: CSIHnDAPm = 1/0 or 0/1)

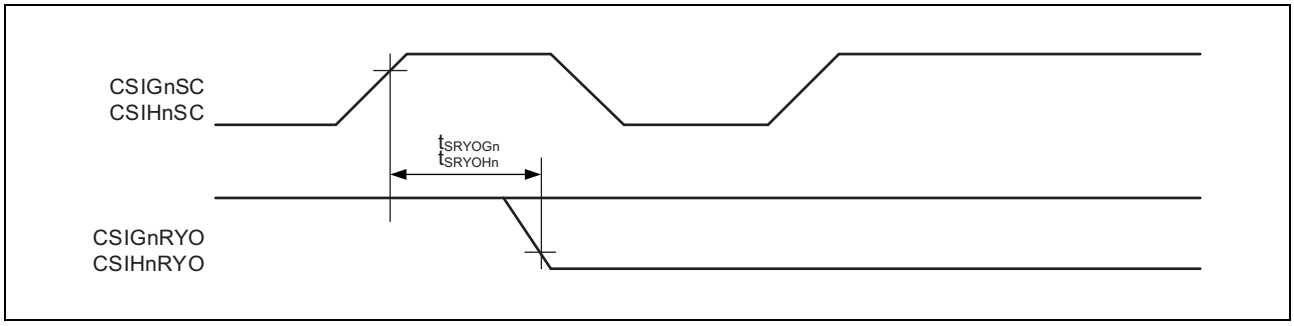


### (5) RYO

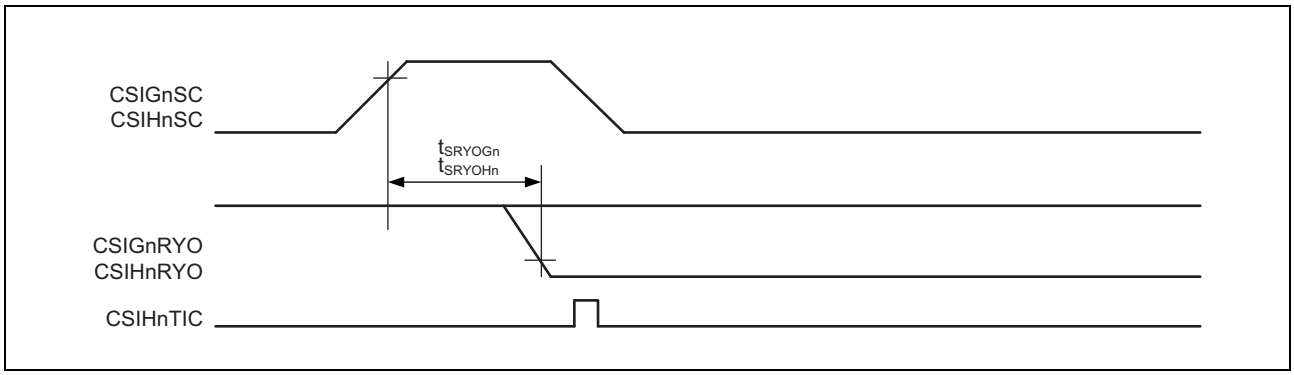
- CSIG (CSIGnCTL1: CSIGnCKR/CSIGnCFG0: CSIGnDAP0 = 0/0)
- CSIH (CSIHnCFGm: CSIHnCKPm/CSIHnCFGm: CSIHnDAPm = 0/0)



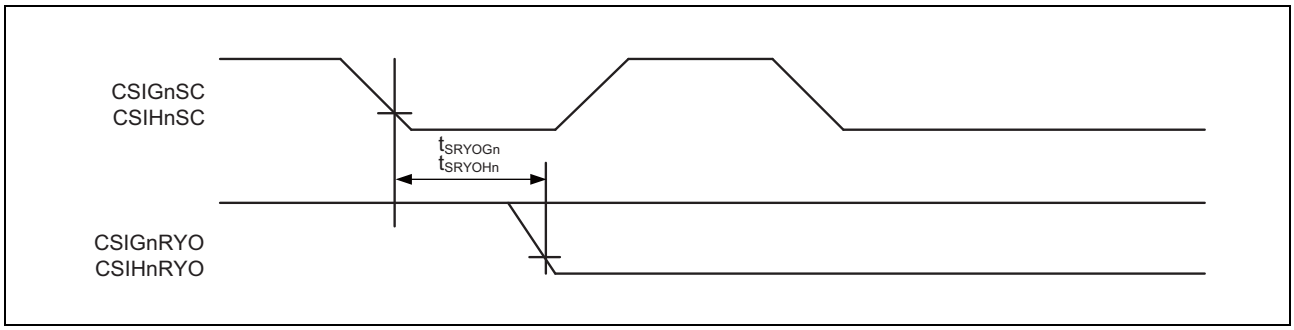

- CSIG (CSIGnCTL1: CSIGnCKR/CSIGnCFG0: CSIGnDAP0 = 0/1)
- CSIH (CSIHnCFGm: CSIHnCKPm/CSIHnCFGm: CSIHnDAPm = 0/1)



- CSIG (CSIGnCTL1: CSIGnCKR/CSIGnCFG0: CSIGnDAP0 = 1/0)
- CSIH (CSIHnCFGm: CSIHnCKPm/CSIHnCFGm: CSIHnDAPm = 1/0)



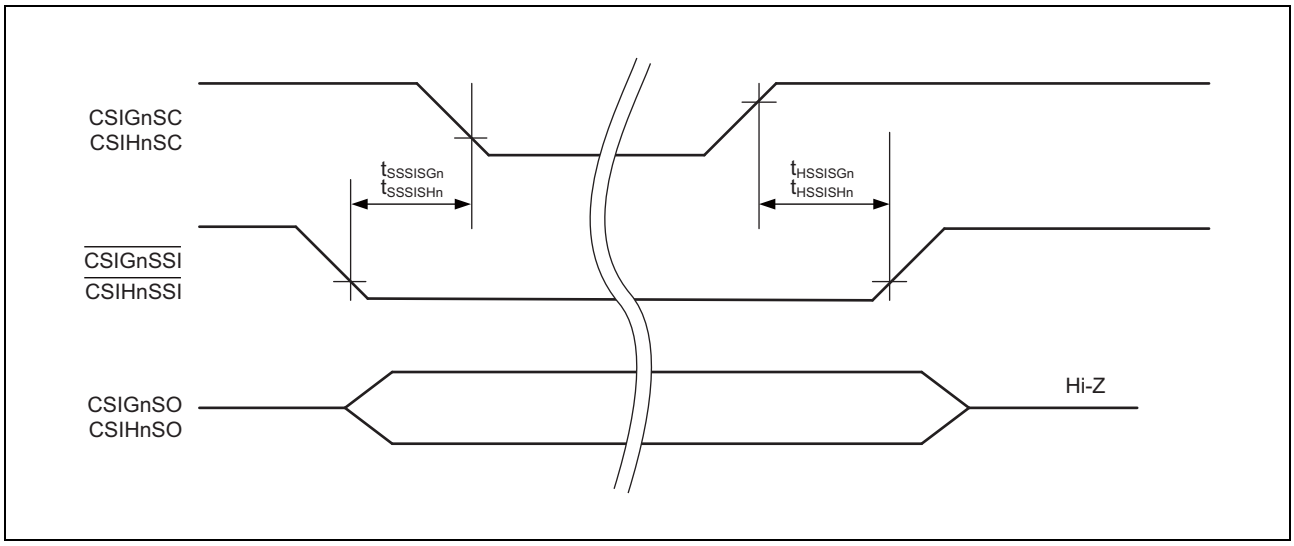

- CSIG (CSIGnCTL1: CSIGnCKR/CSIGnCFG0: CSIGnDAP0 = 1/1)
- CSIH (CSIHnCFGm: CSIHnCKPm/CSIHnCFGm: CSIHnDAPm = 1/1)



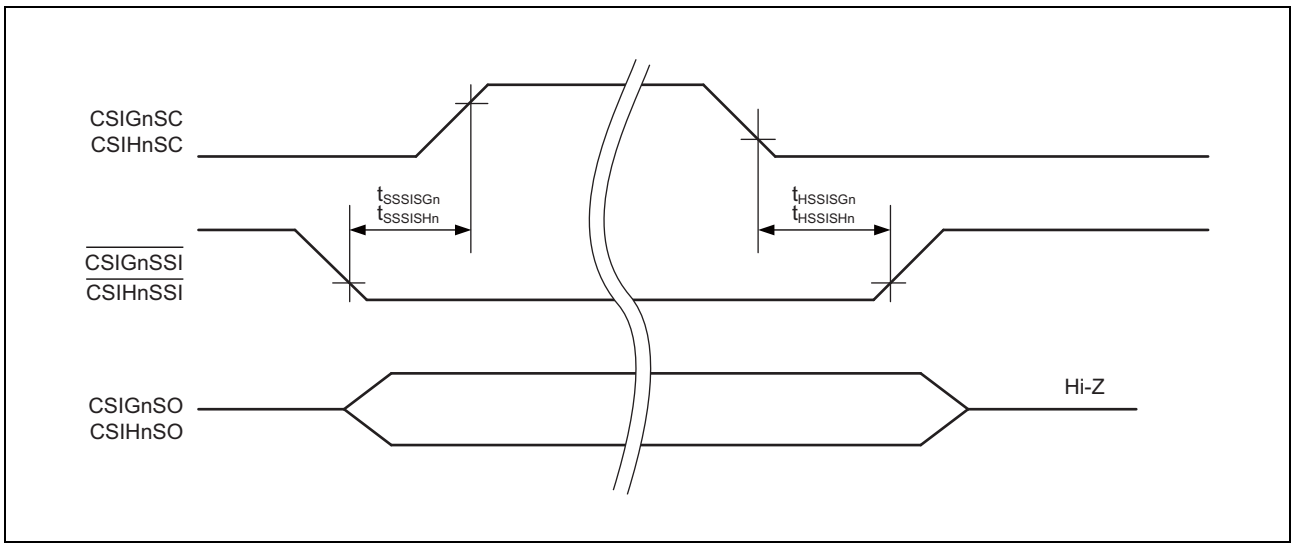
### (6) SSI

**Slave mode:**

- CSIG (CSIGnCTL1: CSIGnSSE=1, CSIGnCTL1: CSIGnCKR/CSIGnCFG0: CSIGnDAP0 = 0/0 or 1/1)
- CSIH (CSIHnCTL1: CSIHnSSE=1, CSIHnCFGm: CSIHnCKPm/CSIHnCFGm: CSIHnDAPm = 0/0 or 1/1)




- CSIG (CSIGnCTL1: CSIGnSSE=1, CSIGnCTL1: CSIGnCKR/CSIGnCFG0: CSIGnDAP0 = 1/0 or 0/1)
- CSIH (CSIHnCTL1: CSIHnSSE=1, CSIHnCFGm: CSIHnCKPm/CSIHnCFGm: CSIHnDAPm = 1/0 or 0/1)



## 40.20 RIIC Timing

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C

**Table 40.12 RIIC Timing (Normal Mode)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| RIIC0SCL clock period | $f_{CLK}$ | | | | 100 | kHz |
| Bus free time (between stop/start condition) | $t_{BUF}$ | | 4.7 | | | µs |
| Hold time*1 | $t_{HD}$: STA | | 4.0 | | | µs |
| RIIC0SCL clock low-level width | $t_{LOW}$ | | 4.7 | | | µs |
| RIIC0SCL clock high-level time | $t_{HIGH}$ | | 4.0 | | | µs |
| Setup time for start/restart condition | $t_{SU}$: STA | | 4.7 | | | µs |
| Data hold time | $t_{HD}$: DAT | CBUS compatible master | 5.0 | | | µs |
| | | $I^2C$ mode | 0*2 | | | µs |
| Data setup time | $t_{SU}$: DAT | | 250 | | | ns |
| Stop condition setup time | $t_{SU}$: STO | | 4.0 | | | µs |
| Capacitance load of each bus line | Cb | | | | 400 | pF |

Note 1. At the start condition, the first clock pulse is generated after the hold time.

Note 2. The system requires a minimum of 300 ns hold time internally for the RIIC0SDA signal (at VIH min. of RIIC0SCL signal). In order to occupy the undefined area at the falling edge of RIIC0SCL.

**Note:** If the system does not extend the RIIC0SCL signal low hold time ($t_{Low}$), only the maximum data hold time ($t_{HD}$: DAT) needs to be satisfied.

**Table 40.13 RIIC Timing (Fast Mode)**

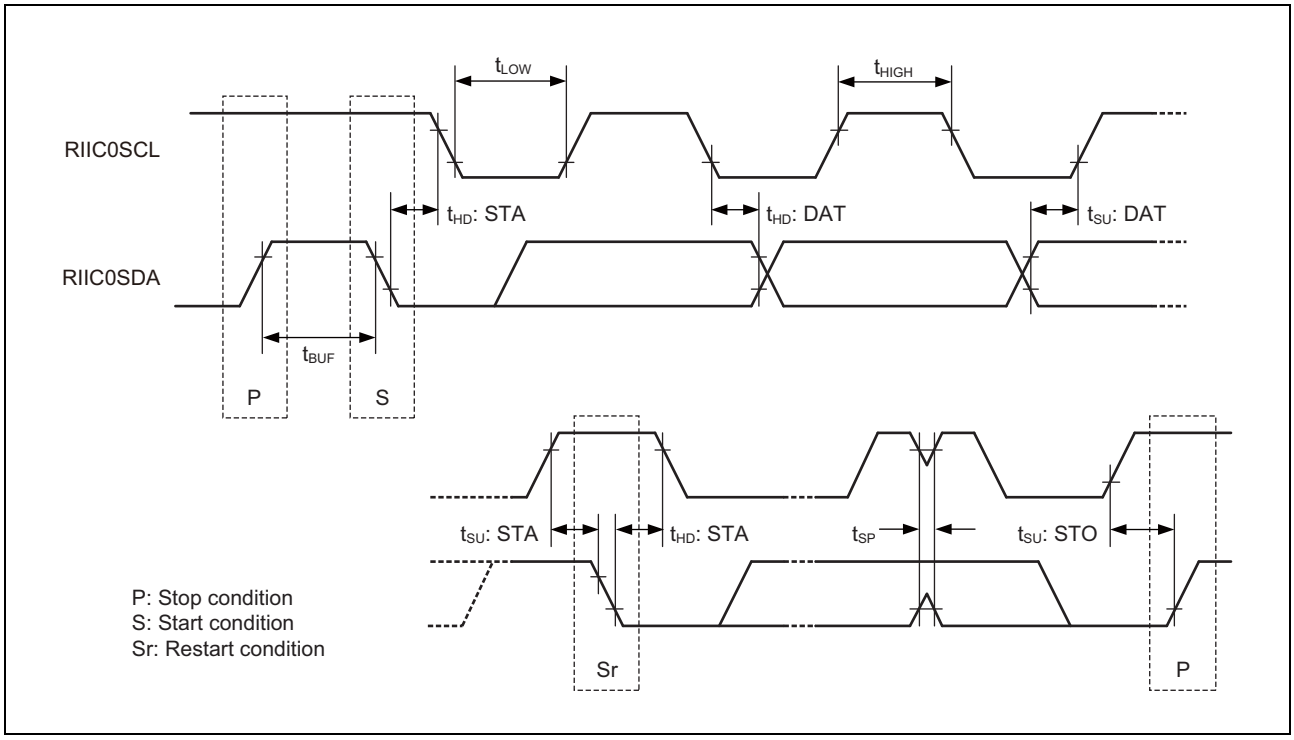
| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| RIIC0SCL clock period | $f_{CLK}$ | | | | 400 | kHz |
| Bus free time (between stop/start condition) | $t_{BUF}$ | | 1.3 | | | µs |
| Hold time*1 | $t_{HD}$: STA | | 0.6 | | | µs |
| RIIC0SCL clock low-level width | $t_{LOW}$ | | 1.3 | | | µs |
| RIIC0SCL clock high-level time | $t_{HIGH}$ | | 0.6 | | | µs |
| Setup time for start/restart condition | $t_{SU}$: STA | | 0.6 | | | µs |
| Data hold time | $t_{HD}$: DAT | $I^2C$ mode | 0*2 | | | µs |
| Data setup time | $t_{SU}$: DAT | | 100*3 | | | ns |
| Stop condition setup time | $t_{SU}$: STO | | 0.6 | | | µs |
| Pulse width with spike suppressed by input filter | $t_{SP}$ | | 0 | | 50 | ns |
| Capacitance load of each bus line | Cb | | | | 400 | pF |

Note 1. At the start condition, the first clock pulse is generated after the hold time.

Note 2. The system requires a minimum of 300 ns hold time internally for the RIIC0SDA signal (at VIH min. of RIIC0SCL signal). In order to occupy the undefined area at the falling edge of RIIC0SCL.

Note 3. The fast mode $I^2C$ bus can be used in normal mode $I^2C$ bus system. In this case, set the fast mode $I^2C$ bus so that it meets the following conditions.
- If the system does not extend the RIIC0SCL signal's low state hold time: $t_{SU}$: DAT ≥ 250 ns
- If the system extends the RIIC0SCL signal's low state hold time:
Transmit the following data bit to the RIIC0SDA line prior to releasing the RIIC0SCL line (1250 ns: Normal mode $I^2C$ bus specification).

**Note:** If the system does not extend the RIIC0SCL signal low hold time ($t_{Low}$), only the maximum data hold time ($t_{HD}$: DAT) needs to be satisfied.

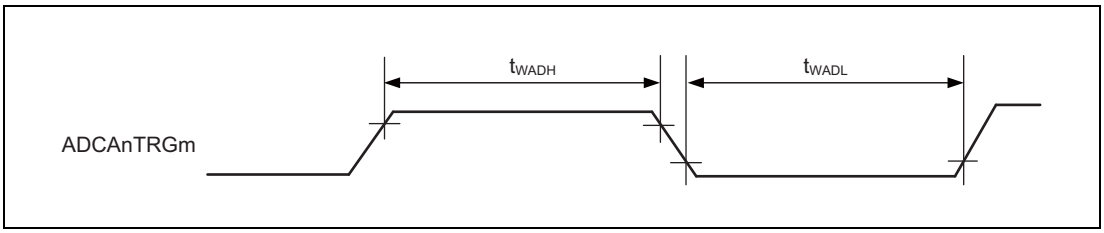

RIIC0SCL
RIIC0SDA
tLOW
tHIGH
tHD: STA
tHD: DAT
tSU: DAT
tBUF
P
S
tSU: STA
tHD: STA
tSP
tSU: STO
Sr
P
P: Stop condition
S: Start condition
Sr: Restart condition

## 40.21 ADTRG Timing

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| ADCAnTRGm input high/ low level width | $t_{WADH}$/ $t_{WADL}$ | | k × Tsamp + 20*1 | | | ns |

Note 1. k: Sampling number of the digital noise filter for each input.
Tsamp: Sampling time of the digital noise filter for each input.

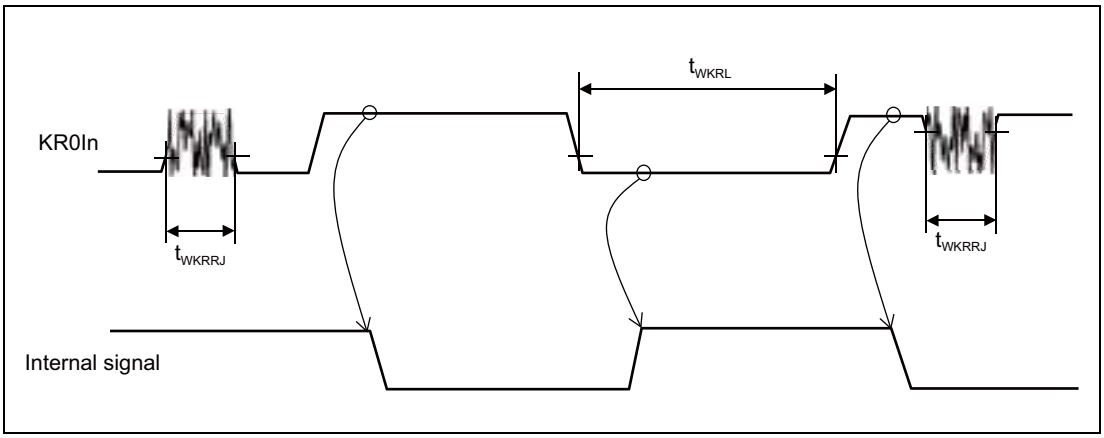



## 40.22 Key Return Timing

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| KR0In input low level width*1 | $t_{WKRL}$ | | 600 | | | ns |
| KR0In pulse rejection*2 | $t_{WKRRJ}$ | | 100 | | | ns |

Note 1. KR0In input width is needed to ensure that the internal key input signal is activated.
Note 2. Pulses shorter than this minimum is ignored. This is reference value.
Noise such as the figure can be filtered.



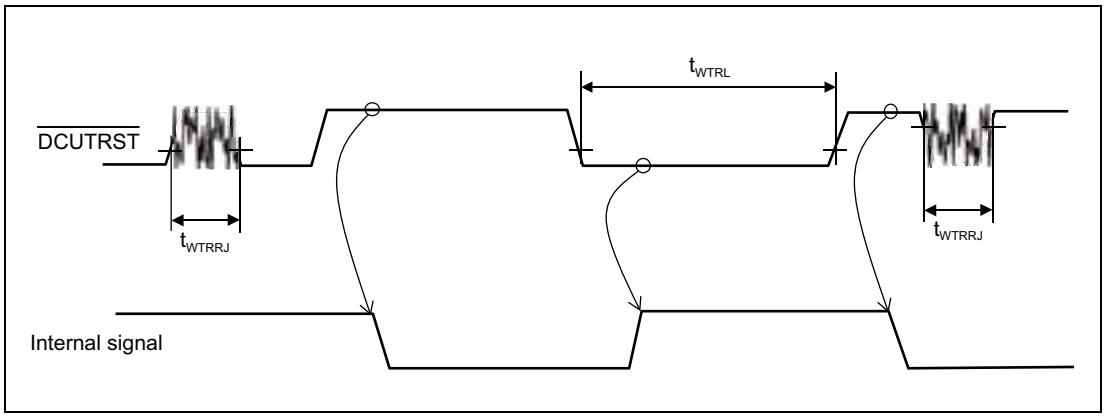
## 40.23 $\overline{\text{DCUTRST}}$ Timing

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{DCUTRST}}$ input low level width*1 | $t_{WTRL}$ | | 600 | | | ns |
| $\overline{\text{DCUTRST}}$ pulse rejection*2 | $t_{WTRRJ}$ | | 100 | | | ns |

Note 1. $\overline{\text{DCUTRST}}$ input width is needed to ensure that the internal DCU reset input signal is activated.

Note 2. Pulses shorter than this minimum is ignored. This is reference value.
Noise such as the figure can be filtered.



# 40.24 Debug Interface Characteristics

## 40.24.1 Nexus Interface Timing

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

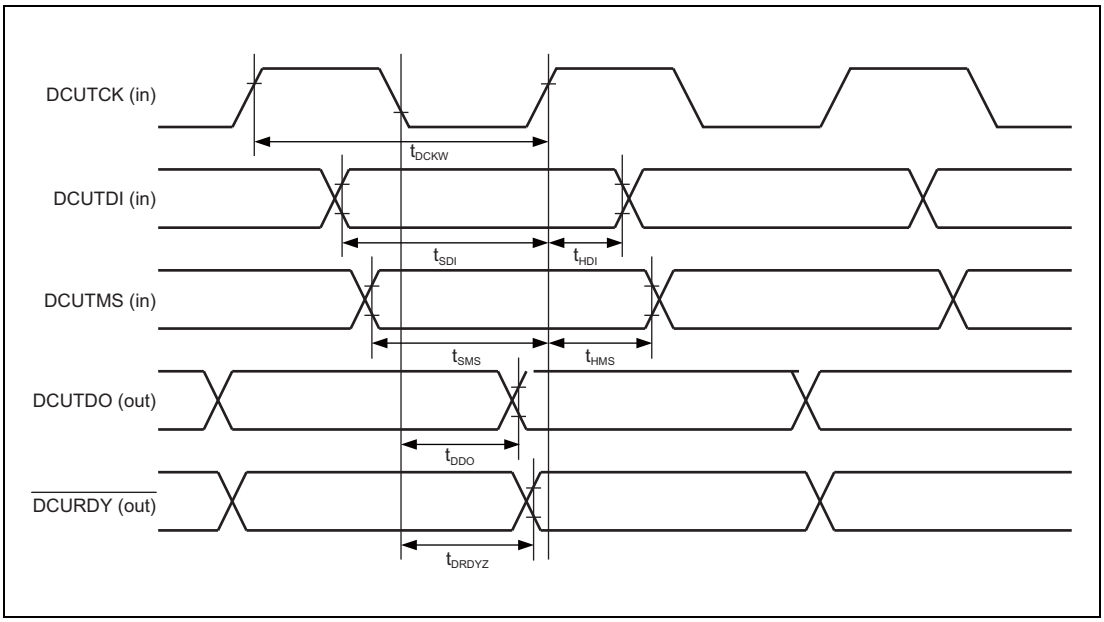
<Input buffer>

DCUTDI, DCUTCK, DCUTMS, $\overline{\text{DCUTRST}}$: TTL

<Output driver strength>

DCUTDO, $\overline{\text{DCURDY}}$: Fast mode

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| DCUTCK cycle width | $t_{DCKW}$ | | 50 | | | ns |
| DCUTDI setup time (vs DCUTCK ↑) | $t_{SDI}$ | | 12 | | | ns |
| DCUTDI hold time (vs DCUTCK ↑) | $t_{HDI}$ | | 3 | | | ns |
| DCUTMS setup time (vs DCUTCK ↑) | $t_{SMS}$ | | 12 | | | ns |
| DCUTMS hold time (vs DCUTCK ↑) | $t_{HMS}$ | | 3 | | | ns |
| DCUTDO delay time (↓ DCUTCK) | $t_{DDO}$ | | 0 | | 20 | ns |
| $\overline{\text{DCURDY}}$ delay time (↓ DCUTCK) | $t_{RDYZ}$ | | 0 | | 20 | ns |



## 40.24.2 LPD (4 pin) Interface Timing

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 100 pF

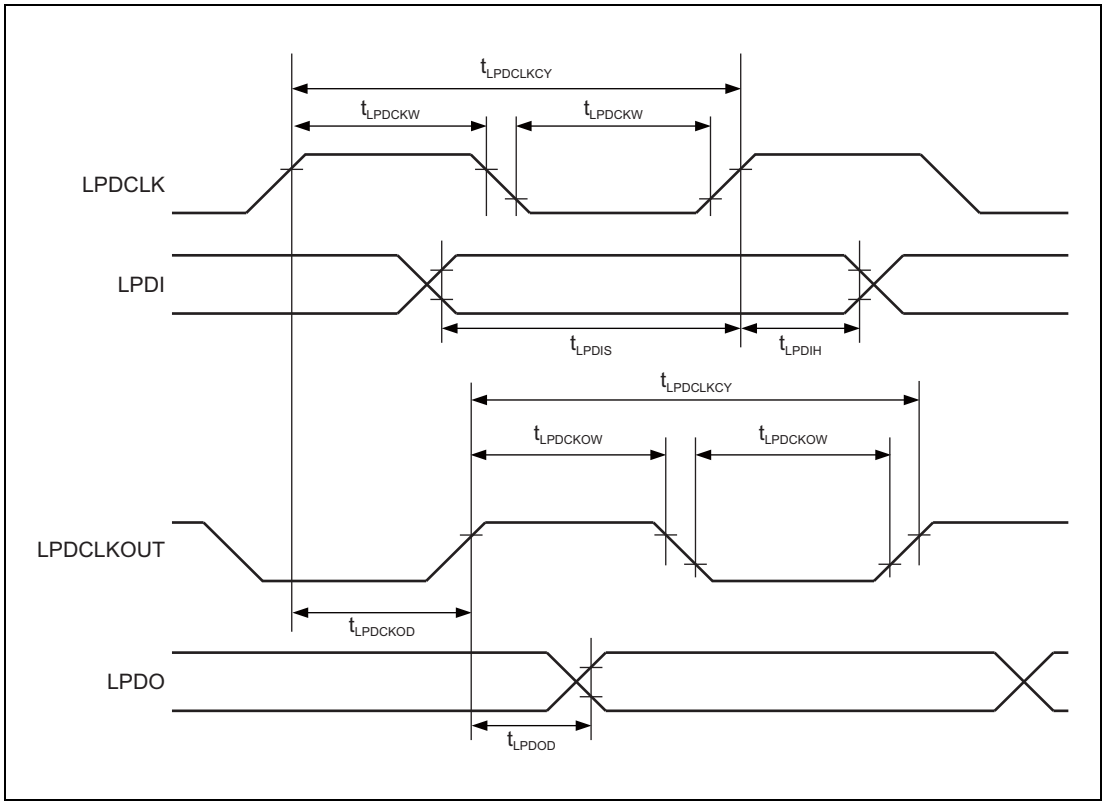
<Input buffer>

LPDCLK, LPDI: TTL

<Output driver strength>

LPDCLKOUT, LPDO: Fast mode

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| LPDCLK cycle time/ LPDCLKOUT cycle time | $t_{LPDCLKCY}$ | | 83.3 (max.12MHz) | | | ns |
| LPDCLK High-level width/ LPDCLK Low-level width | $t_{LPDCKW}$ | | 0.5 × $t_{LPDCLKCY}$ – 10 | | | ns |
| LPDCLKOUT High-level width/ LPDCLKOUT low-level width | $t_{LPDCKOW}$ | | $t_{LPDCKW}$ – 10 | | | ns |
| LPDI setup time (LPDCLK ↑) | $t_{LPDIS}$ | | 41 | | | ns |
| LPDI hold time (LPDCLK ↑) | $t_{LPDIH}$ | | 3 | | | ns |
| LPDCLK to LPDCLKOUT delay time | $t_{LPDCKOD}$ | | | | 44 | ns |
| LPDO delay time (LPDCLKOUT ↑) | $t_{LPDOD}$ | | 0 | | 15 | ns |



### 40.24.3 LPD (1 pin) Interface Timing

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 50 pF

<Input buffer>

LPDIO: TTL

<Output driver strength>

LPDIO: Fast mode

<External pull-up resistor>

LPDIO: 4.7 kΩ to 10 kΩ

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| LPD (1 pin) Baud rate | | | | | 2.0 | Mbps |

# 40.25 Flash Programming Characteristics

## 40.25.1 Code Flash

The code flash memory is shipped in the erased state. If the code flash memory is read where it has not been written after erasure (no write condition), an ECC error is generated, resulting in the occurrence of an exception.

**Condition:** REGVCC = EVCC = VPOC to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

**Table 40.14 Basic characteristics**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Operation frequency | $f_{PCLK}$*3 | ECO | 4*4 | | 20 | MHz |
| | | ADVANCED, PREMIUM | 4*4 | | 30 | MHz |
| Number of rewrites*1 | CWRT | Data retention of 20 years*2 | 1000 | | | times |

Note 1. The number of rewrites is the number of erasures for each block. When the number of rewrites is “n” (n = 1000), the device can be erased “n” times for each block. For example, when a block of 32 KB is erased after 256 bytes of writing have been performed for different addresses 128 times, the number of rewrites is counted as 1. However, multiple writing to the same address is not possible with 1 erasure (overwriting prohibited).

Note 2. Retention period under average Ta = 85°C. This is the period starting on completion of a successful erasure of the code flash memory.

Note 3. $f_{PCLK}$ = 1/4 $f_{CPUCLK}$: System operating frequency for internal flash.

Note 4. Only for program/erase operation.

**Table 40.15 Programming characteristic (1/2)**

| Item | Symbol | Condition | Block size | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Programming time | | $f_{PCLK}$ ≥ 20 MHz<br>CWRT < 100 times | 256 B | | 0.4*1 | 6*1 | ms |
| | | | 8 KB | | 20 | 90 | ms |
| | | | 32 KB | | 80 | 360 | ms |
| | | | 256 KB | | 0.6 | 2.7 | s |
| | | | 384 KB | | 0.9 | 4.1 | s |
| | | | 512 KB | | 1.2 | 5.4 | s |
| | | | 768 KB | | 1.7 | 8.1 | s |
| | | | 1 MB | | 2.3 | 10.8 | s |
| | | | 1.5 MB | | 3.4 | 16.2 | s |
| | | | 2 MB | | 4.5 | 21.5 | s |
| | | $f_{PCLK}$ ≥ 20 MHz<br>CWRT ≥ 100 times | 256 B | | 0.5*1 | 7.2*1 | ms |
| | | | 8 KB | | 24 | 108 | ms |
| | | | 32 KB | | 96 | 432 | ms |
| | | | 256 KB | | 0.7 | 3.3 | s |
| | | | 384 KB | | 1.1 | 4.9 | s |
| | | | 512 KB | | 1.4 | 6.5 | s |
| | | | 768 KB | | 2.1 | 9.8 | s |
| | | | 1 MB | | 2.7 | 13 | s |
| | | | 1.5 MB | | 4.1 | 19.5 | s |
| | | | 2 MB | | 5.4 | 26 | s |

**Table 40.15 Programming characteristic (2/2)**

| Item | Symbol | Condition | Block size | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Erase time | | $f_{PCLK}$ ≥ 20 MHz<br>CWRT < 100 times | 8 KB | | 39 | 120 | ms |
| | | | 32 KB | | 141 | 480 | ms |
| | | | 256 KB | | 1.2 | 3.5 | s |
| | | | 384 KB | | 1.7 | 5.3 | s |
| | | | 512 KB | | 2.3 | 7 | s |
| | | | 768 KB | | 3.4 | 10.5 | s |
| | | | 1 MB | | 4.5 | 14 | s |
| | | | 1.5 MB | | 6.8 | 21 | s |
| | | | 2 MB | | 9 | 28 | s |
| | | $f_{PCLK}$ ≥ 20 MHz<br>CWRT ≥ 100 times | 8 KB | | 47 | 144 | ms |
| | | | 32 KB | | 169 | 576 | ms |
| | | | 256 KB | | 1.4 | 4.2 | s |
| | | | 384 KB | | 2.1 | 6.3 | s |
| | | | 512 KB | | 2.7 | 8.4 | s |
| | | | 768 KB | | 4.1 | 12.6 | s |
| | | | 1 MB | | 5.4 | 16.8 | s |
| | | | 1.5 MB | | 8.1 | 25.2 | s |
| | | | 2 MB | | 10.8 | 33.6 | s |

Note 1. Only the processing time of the hardware. The overhead required by the software is not included.

## 40.25.2 Data Flash

The data flash memory is shipped in the erased state. If the data flash memory is read where it has not been written after erasure (no write condition), an ECC error is generated, resulting in the occurrence of an exception

**Condition:** REGVCC = EVCC = VPOC to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

**Table 40.16 Basic characteristics**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Operation frequency | $f_{PCLK}$*3 | ECO | 4*4 | | 20 | MHz |
| | | ADVANCED, PREMIUM | 4*4 | | 30 | MHz |
| Number of rewrites*1 | CWRT | Data retention 20 years*2 | 125 k | | | times |
| | | Data retention 3 years*2 | 250 k | | | times |

Note 1. The number of rewrites is the number of erasures for each block. When the number of rewrites is “n” (n = 125000), the device can be erased “n” times for each block. For example, when a block of 64 bytes is erased after 4 bytes of writing have been performed for different addresses 168 times, the number of rewrites is counted as 1. However, multiple writing to the same address is not possible with 1 erasure (overwriting prohibited).

Note 2. Retention period under average Ta = 85°C. This is the period starting on completion of a successful erasure of the data flash memory.

Note 3. $f_{PCLK}$ = 1/4 $f_{CPUCLK}$: System operating frequency for internal flash.

Note 4. Only for program/erase operation.

**Table 40.17 Programming characteristics**

| Item | Symbol | Condition | Block size | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Programming time | | $f_{PCLK}$ ≥ 20 MHz | 4 B | | 0.16*1 | 1.7*1 | ms |
| | | $f_{PCLK}$ ≥ 20 MHz | 32 KB | | 1.4 | 6.8 | s |
| | | $f_{PCLK}$ ≥ 20 MHz | 64 KB | | 2.79 | 13.44 | s |
| Erase time | | $f_{PCLK}$ ≥ 20 MHz | 64 B | | 1.7*1 | 10*1 | ms |
| | | $f_{PCLK}$ ≥ 20 MHz | 32 KB | | 0.9 | 5.2 | s |
| | | $f_{PCLK}$ ≥ 20 MHz | 64 KB | | 1.74 | 10.24 | s |
| Blank check time | | $f_{PCLK}$ ≥ 20 MHz | 4 B | | | 30*1 | µs |
| | | | 64 B | | | 100*1 | µs |
| | | | 32 KB | | | 35.2 | ms |
| | | | 64 KB | | | 70.4 | ms |

Note 1. Only the processing time of the hardware. The overhead required by the software is not included.

## 40.25.3 Serial Programming Interface

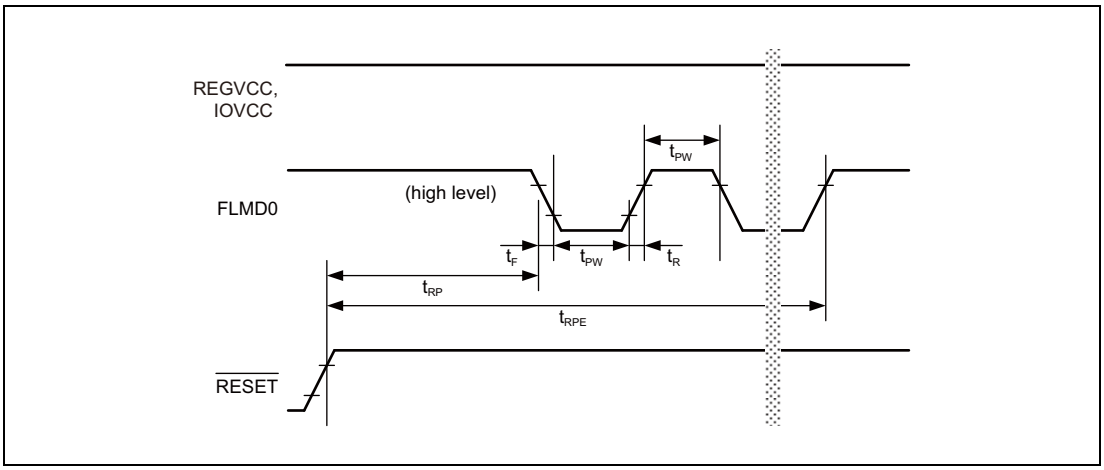
### 40.25.3.1 Serial Programmer Setup Timing

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| FLMD0 pulse input start time | $t_{RP}$ | | 1.5 | | | ms |
| FLMD0 pulse input end time | $t_{RPE}$ | | | | 101.5 | ms |
| FLMD0 low/high level width | $t_{PW}$ | | 1.6 | | | µs |
| FLMD0 rise time | $t_R$ | | | | 20 | ns |
| FLMD0 fall time | $t_F$ | | | | 20 | ns |

**NOTE**

IOVCC: EVCC = A0VREF = A1VREF



### 40.25.3.2 Flash Programming Interface

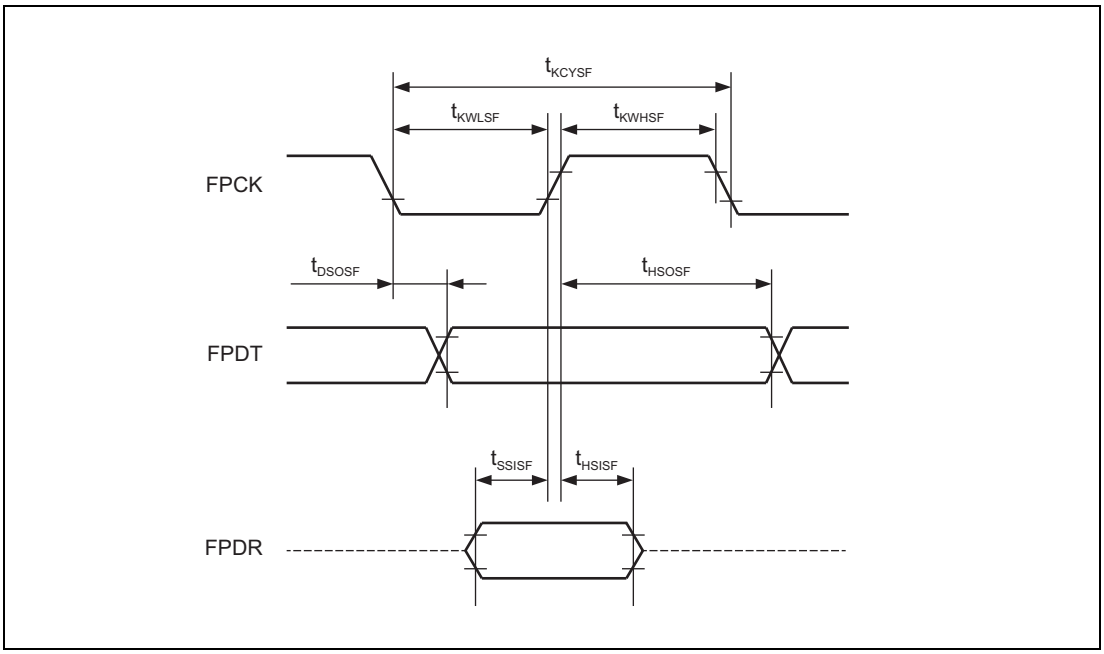
**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Flash Programming transfer rate | | 1-wired UART mode | | | 1 | Mbps |
| | | 2-wired UART mode (ECO) | | | 1 | Mbps |
| | | 2-wired UART mode (PREMIUM, ADVANCED) | | | 1.5 | Mbps |
| FPCK cycle time | $t_{KCYSF}$ | 3-wired clock sync mode | 200*1 | | | ns |
| FPCK high level width | $t_{KWHSF}$ | 3-wired clock sync mode | $t_{KCYSF}$ / 2 – 15 | | | ns |
| FPCK low level width | $t_{KWLSF}$ | 3-wired clock sync mode | $t_{KCYSF}$ / 2 – 15 | | | ns |
| FPDR setup time (vs. FPCK) | $t_{SSISF}$ | 3-wired clock sync mode | $t_{Pcyc}$ × 2 | | | ns |
| FPDR hold time (vs. FPCK) | $t_{HSISF}$ | 3-wired clock sync mode | $t_{Pcyc}$ × 2 | | | ns |
| FPDT output delay (vs. FPCK) | $t_{DSOSF}$ | 3-wired clock sync mode Not continuous transfer (data: 1st bit) | | | 0 | ns |
| | | 3-wired clock sync mode Not continuous transfer (data: except 1st bit) | | | $–t_{KWHSF}$ + 3 × $t_{Pcyc}$ + 36 | ns |
| FPDT hold time (vs. FPCK) | $t_{HSOSF}$ | 3-wired clock sync mode | $t_{Pcyc}$ × 2 | | | ns |

Note 1. Input an external clock that is more than 6 clocks of PCLK.

**NOTE**

$t_{Pcyc}$ is period of PCLK.



## 40.26 A/D Converter Characteristics

**Condition:** REGVCC = EVCC = 3.0 V to 5.5 V, A0VREF = 3.0 V to 5.5 V,
A1VREF = 3.0 V to 5.5 V, AWOVSS = ISOVSS = EVSS = A0VSS = A1VSS = 0 V,
CAWOVCL: 0.1µF ±30%, CISOVCL: 0.1µF ±30%, Tj = –40 to (depend on the product) °C,
CL = 30 pF

(1/2)

| Item | Symbol | Condition | | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|---|
| Conversion clock | ADCLKn | | | | 8*3 | | 40 | MHz |
| Resolution | RESn | 12-bit mode | | | 12 | 12 | 12 | bit |
| | | 10-bit mode | | | 10 | 10 | 10 | bit |
| Conversion time | $t_{CONn}$ | ADCAnSMPCR.SMPT[7:0] = 12 H(40 cycle) (8M Hz*3 ≤ ADCLKn ≤ 32 MHz), External MPX is not used | | | 1.25 | | 5 | µs |
| | | ADCAnSMPCR.SMPT[7:0] = 18 H (46 cycle) (8 MHz*3 ≤ ADCLKn ≤ 40 MHz), External MPX is not used | | | 1.15 | | 5.75 | µs |
| | | ADCAnSMPCR.SMPT[7:0]=12 H (80 cycle) (8 MHz*3 ≤ ADCLKn ≤ 32 MHz), External MPX is used | | | 2.5*4 | | 10 | µs |
| | | ADCAnSMPCR.SMPT[7:0] = 18 H (92 cycle) (8 MHz*3 ≤ ADCLKn ≤ 40 MHz), External MPX is used | | | 2.3*4 | | 11.5 | µs |
| Sampling time | $t_{SMP}$ | ADCAnSMPCR.SMPT[7:0] = 12 H (18 cycle) (8 MHz*3 ≤ ADCLKn ≤ 32 MHz) | | | 0.56 | | 2.25 | µs |
| | | ADCAnSMPCR.SMPT[7:0] = 18 H (24cycle) (8 MHz*3 ≤ ADCLKn ≤ 40 MHz) | | | 0.6 | | 3 | µs |
| Overall error*1 | TOEn | 12-bit mode | AnVREF = 4.5 V to 5.5 V | ADCAnIm (w/o T&H) | | | ±4.0 | LSB |
| | | | | ADCA0I0-5 (w/ T&H) | | | ±6.0 | LSB |
| | | | AnVREF = 3.0 V to 4.5 V | ADCAnIm (w/o T&H) | | | ±6.0 | LSB |
| | | | | ADCA0I0-5 (w/ T&H) | | | ±8.0 | LSB |
| | | 10-bit mode | AnVREF = 4.5 V to 5.5 V | ADCAnIm | | | ±1.0 | LSB |
| | | | | ADCAnImS | | | ±2.0 | LSB |
| | | | AnVREF = 3.0 V to 4.5 V | ADCAnIm | | | ±1.5 | LSB |
| | | | | ADCAnImS | | | ±2.5 | LSB |
| Analog input voltage | VAIN0SN | ADCAnIm | T&H not used | | AnVSS | | AnVREF | V |
| | | ADCA0I0-5 | T&H used | | 0.2 | | A0VREF – 0.2 | V |
| | | ADCA0ImS | A0VREF ≥ EVCC | | A0VSS | | EVCC | V |
| | | | A0VREF < EVCC | | A0VSS | | A0VREF | V |
| | | ADCA1ImS | A1VREF ≥ EVCC | | A1VSS | | EVCC | V |
| | | | A1VREF < EVCC | | A1VSS | | A1VREF | V |
| Operation current | IA0VREF IA1VREF | T&H not used | | | | 1.1 | 3.0 | mA |
| | | T&H used (max. 6 pins) | | | | | *2 | mA |
| STOP, DeepSTOP, Cyclic STOP current (@LPS is stopped) | IA0VREFS IA1VREFS | | | | | 1 | 10 | µA |
| T&H current | ITH | | | | | 0.5 | 1.3 | mA/ch |
| T&H sampling time | $t_{THSMP}$ | | | | 450 | | | ns |
| T&H hold time | $t_{THHOLD}$ | | | | | | 10 | µs |
| Set up time of self diagnosis voltage circuit | $t_{BOOT}$ | | | | 500 | | | ns |
| Set up time of self diagnosis voltage level | $t_{OUT}$ | | | | 500 | | | ns |
| Pull-down resistor for discharge mode | | ADCnIm pins | | | 350 | 500 | 650 | kΩ |
| | | ADCnImS pins | | | 100 | 215 | 800 | kΩ |

(2/2)

| Item | Symbol | Condition | | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|---|
| Accuracy of self-diagnosis function | TESH0SN | 12bit mode | Self-diagnosis voltage level = AnVREF | | 4015-\|TOEn\| | | 4095 | — |
| | | | Self-diagnosis voltage level = 2/3AnVREF | | 2651-\|TOEn\| | 2731 | 2811+\|TOEn\| | — |
| | | | Self-diagnosis voltage level = 1/2AnVREF | | 1968-\|TOEn\| | 2048 | 2128+\|TOEn\| | — |
| | | | Self-diagnosis voltage level = 1/3AnVREF | | 1285-\|TOEn\| | 1365 | 1445+\|TOEn\| | — |
| | | | Self-diagnosis voltage level = AnVSS | | 0 | | 80+\|TOEn\| | — |
| | | 10bit mode | Self-diagnosis voltage level = AnVREF | | 1003-\|TOEn\| | | 1023 | — |
| | | | Self-diagnosis voltage level = 2/3AnVREF | | 663-\|TOEn\| | 683 | 703+\|TOEn\| | — |
| | | | Self-diagnosis voltage level = 1/2AnVREF | | 492-\|TOEn\| | 512 | 532+\|TOEn\| | — |
| | | | Self-diagnosis voltage level = 1/3AnVREF | | 321-\|TOEn\| | 341 | 361+\|TOEn\| | — |
| | | | Self-diagnosis voltage level = AnVSS | | 0 | | 20+\|TOEn\| | — |
| Integral nonlinearity error*1 | ILEn | 12-bit mode | AnVREF = 4.5 V to 5.5 V | ADCAnIm (w/o T&H) | | | ±2.0 | LSB |
| | | | | ADCA0I0-5 (w/ T&H) | | | ±3.0 | LSB |
| | | | AnVREF = 3.0 V to 4.5 V | ADCAnIm (w/o T&H) | | | ±3.0 | LSB |
| | | | | ADCA0I0-5 (w/ T&H) | | | ±4.0 | LSB |
| | | 10-bit mode | AnVREF = 4.5 V to 5.5 V | ADCAnIm | | | ±1.0 | LSB |
| | | | | ADCAnImS | | | ±2.0 | LSB |
| | | | AnVREF = 3.0 V to 4.5 V | ADCAnIm | | | ±1.5 | LSB |
| | | | | ADCAnImS | | | ±2.5 | LSB |
| Differential nonlinearity error*1 | DLEn | 12-bit mode | AnVREF = 4.5 V to 5.5 V | ADCAnIm (w/o T&H) | | | ±1.0 | LSB |
| | | | | ADCA0I0-5 (w/ T&H) | | | ±2.0 | LSB |
| | | | AnVREF = 3.0 V to 4.5 V | ADCAnIm (w/o T&H) | | | ±3.0 | LSB |
| | | | | ADCA0I0-5 (w/ T&H) | | | ±4.0 | LSB |
| | | 10-bit mode | AnVREF = 4.5 V to 5.5 V | ADCAnIm | | | ±1.0 | LSB |
| | | | | ADCAnImS | | | ±1.5 | LSB |
| | | | AnVREF = 3.0V to 4.5V | ADCAnIm | | | ±1.0 | LSB |
| | | | | ADCAnImS | | | ±2.0 | LSB |
| Zero scale error (offset error)*1 | ZSEn | 12-bit mode | AnVREF = 4.5 V to 5.5 V | ADCAnIm (w/o T&H) | | | ±3.5 | LSB |
| | | | | ADCA0I0-5 (w/ T&H) | | | ±5.5 | LSB |
| | | | AnVREF = 3.0 V to 4.5 V | ADCAnIm (w/o T&H) | | | ±5.5 | LSB |
| | | | | ADCA0I0-5 (w/ T&H) | | | ±7.5 | LSB |
| | | 10-bit mode | AnVREF = 4.5 V to 5.5 V | ADCAnIm | | | ±0.5 | LSB |
| | | | | ADCAnImS | | | ±1.5 | LSB |
| | | | AnVREF = 3.0 V to 4.5 V | ADCAnIm | | | ±1.0 | LSB |
| | | | | ADCAnImS | | | ±2.0 | LSB |
| Full scale error*1 | FSEn | 12-bit mode | AnVREF = 4.5 V to 5.5 V | ADCAnIm (w/o T&H) | | | ±3.5 | LSB |
| | | | | ADCA0I0-5 (w/ T&H) | | | ±5.5 | LSB |
| | | | AnVREF = 3.0 V to 4.5 V | ADCAnIm (w/o T&H) | | | ±5.5 | LSB |
| | | | | ADCA0I0-5 (w/ T&H) | | | ±7.5 | LSB |
| | | 10-bit mode | AnVREF = 4.5 V to 5.5 V | ADCAnIm | | | ±0.5 | LSB |
| | | | | ADCAnImS | | | ±1.5 | LSB |
| | | | AnVREF = 3.0 V to 4.5 V | ADCAnIm | | | ±1.0 | LSB |
| | | | | ADCAnImS | | | ±2.0 | LSB |

Note 1. This does not include quantization error.

Note 2. 3.0 + 1.3 × (the number of used T&H)

Note 3. Include the oscillation accuracy of HS IntOSC.

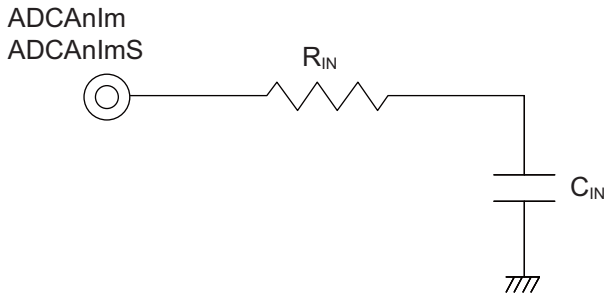
Note 4. When the external multiplexer is used, the detailed time of A/D conversion is MPX setup time, sampling time and successive approximation time. MPX setup time is same as “sampling time + successive approximation time”.
Conversion precision include the influence of the injection current.

**Note:** Conversion accuracy when ADCA0ImS terminal is converted in 12-bit mode: Conversion accuracy can be applied if lower 2-bit is ignored from conversion result.

**CAUTION**

When an external digital pulse is applied to AP0, AP1, P8, P9 and P18 pins during an A/D conversion this may lead to an A/D conversion result with a larger conversion error as expected due to the coupling noise of the external digital pulse.
The same behavior may apply when the digital buffer is used as output pin. For the output port the potential degradation increases with the driven total output current of the port. In addition the conversion resolution may drop if the output current fluctuates at adjacent pins due to the coupling effect of the external circuit connected to these port pins.

### 40.26.1 Equivalent Circuit of the Analog Input Block




| Terminals | Condition | $R_{IN}$ (kΩ) | $C_{IN}$ (pF) |
|---|---|---|---|
| ADCA0I0 to 15 | When T&H is used | 14.0 | 1.8 |
| | When T&H is not used | 3.9 | 2.1 |
| ADCA0I0S to 3S, 5S to 16S | | 5.0 | 8.6 |
| ADCA0I4S, 17S to 19S | — | 5.4 | 8.6 |
| ADCA1I0 to 15 | — | 3.6 | 1.8 |
| ADCA1I0S to 7S | — | 4.7 | 4.5 |

**CAUTION**

This specification is not tested during outgoing inspection. Therefore $R_{IN}$ and $C_{IN}$ are reference values only and not guaranteed. In addition these values are specified as maximum values.

## 40.27 Injection Currents

Table 40.18 Definition of Pin Group (176 pin)

| Symbol | Power Supply for Pin Group | Pin |
|---|---|---|
| PgR | REGVCC, AWOVSS | IP0_0 |
| PgE | EVCC, EVSS | JP0, P0, P1, P2, P10, P11, P12, P20 |
| PgE’ | EVCC, EVSS | P8, P9, P18 |
| PgA0 | A0VREF, A0VSS | AP0 |
| PgA1 | A1VREF, A1VSS | AP1 |

Table 40.19 Definition of Pin Group (144 pin)

| Symbol | Power Supply for Pin Group | Pin |
|---|---|---|
| PgR | REGVCC, AWOVSS | IP0_0 |
| PgE | EVCC, EVSS | JP0, P0, P1, P10, P11, P12, P20 |
| PgE’ | EVCC, EVSS | P8, P9, P18 |
| PgA0 | A0VREF, A0VSS | AP0 |
| PgA1 | A1VREF, A1VSS | AP1 |

Table 40.20 Definition of Pin Group (100 pin)

| Symbol | Power Supply for Pin Group | Pin |
|---|---|---|
| PgE | EVCC, EVSS | JP0, P0, P10, P11 |
| PgE’ | EVCC, EVSS | P8, P9 |
| PgA0 | A0VREF, A0VSS | AP0 |

## 40.27.1 Absolute Maximum Ratings

### 40.27.1.1 176 pin and 144 pin

| Item | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Positive overload current VIN > VCC | $I_{INJPM}$ | PgE | Per pin | | | 10 | mA |
| | | | Total | | | 120 | mA |
| | | PgE’ | Per pin | | | 10 | mA |
| | | | Total | | | 120 | mA |
| | | PgA0 | Per pin | | | 10 | mA |
| | | | Total | | | 60 | mA |
| | | PgA1 | Per pin | | | 10 | mA |
| | | | Total | | | 60 | mA |
| | | PgR | Per pin | | | 10 | mA |
| Negative overload current VIN < VSS | $I_{INJNM}$ | PgE | Per pin | | | –10 | mA |
| | | | Total | | | –120 | mA |
| | | PgE’ | Per pin | | | –10 | mA |
| | | | Total | | | –120 | mA |
| | | PgA0 | Per pin | | | –10 | mA |
| | | | Total | | | –60 | mA |
| | | PgA1 | Per pin | | | –10 | mA |
| | | | Total | | | –60 | mA |
| | | PgR | Per pin | | | –10 | mA |

**CAUTIONS**

1. The DC injection current (Total) must satisfy the specifications of the injection current per pin.
2. In case of injected current for PgA0 and PgA1, TESH0SN cannot be kept. Its deviating value will increase sharply with increasing absolute value of injection current.

### 40.27.1.2 100 pin

| Item | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Positive overload current VIN > VCC | $I_{INJPM}$ | PgE | Per pin | | | 10 | mA |
| | | | Total | | | 60 | mA |
| | | PgE’ | Per pin | | | 10 | mA |
| | | | Total | | | 60 | mA |
| | | PgA0 | Per pin | | | 10 | mA |
| | | | Total | | | 60 | mA |
| Negative overload current VIN < VSS | $I_{INJNM}$ | PgE | Per pin | | | –10 | mA |
| | | | Total | | | –60 | mA |
| | | PgE’ | Per pin | | | –10 | mA |
| | | | Total | | | –60 | mA |
| | | PgA0 | Per pin | | | –10 | mA |
| | | | Total | | | –60 | mA |

**CAUTIONS**

1. The DC injection current (Total) must satisfy the specifications of the injection current per pin.
2. In case of injected current for PgA0, TESH0SN cannot be kept. Its deviating value will increase sharply with increasing absolute value of injection current.

## 40.27.2 DC Characteristics for Overload Current

### 40.27.2.1 176 pin and 144 pin

| Item | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Positive overload current<br>VIN > VCC | $I_{INJP}$ | PgE | Per pin | | | 2 | mA |
| | | | Total | | | 100 | mA |
| | | PgE’ | Per pin | | | 3 | mA |
| | | | Total | | | 40 | mA |
| | | PgA0 | Per pin | | | 3 | mA |
| | | | Total | | | 20 | mA |
| | | PgA1 | Per pin | | | 3 | mA |
| | | | Total | | | 20 | mA |
| | | PgR | Per pin | | | 2 | mA |
| Negative overload current<br>VIN < VSS | $I_{INJN}$ | PgE | Per pin | | | –2 | mA |
| | | | Total | | | –100 | mA |
| | | PgE’ | Per pin | | | –3 | mA |
| | | | Total | | | –40 | mA |
| | | PgA0 | Per pin | | | –3 | mA |
| | | | Total | | | –20 | mA |
| | | PgA1 | Per pin | | | –3 | mA |
| | | | Total | | | –20 | mA |
| | | PgR | Per pin | | | –2 | mA |

**NOTE**

These specifications are not tested on sorting and are specified based on the device characterization.

### 40.27.2.2 100 pin

| Item | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Positive overload current<br>VIN > VCC | $I_{INJP}$ | PgE | Per pin | | | 2 | mA |
| | | | Total | | | 50 | mA |
| | | PgE’ | Per pin | | | 3 | mA |
| | | | Total | | | 20 | mA |
| | | PgA0 | Per pin | | | 3 | mA |
| | | | Total | | | 20 | mA |
| Negative overload current<br>VIN < VSS | $I_{INJN}$ | PgE | Per pin | | | –2 | mA |
| | | | Total | | | –50 | mA |
| | | PgE’ | Per pin | | | –3 | mA |
| | | | Total | | | –20 | mA |
| | | PgA0 | Per pin | | | –3 | mA |
| | | | Total | | | –20 | mA |

**NOTE**

These specifications are not tested on sorting and are specified based on the device characterization.

# 40.28 Thermal Characteristics

## 40.28.1 Parameters

| Package | Item | Symbol | Estimate | Unit | Note |
|---|---|---|---|---|---|
| 176 pin LQFP | Thermal Resistance | Θja | 35.5 | °C/W | Conforming to JESD51-7 (4 layers) |
| | Thermal Characterization Parameter | ψjb | 27.6 | | |
| 144 pin LQFP | Thermal Resistance | Θja | 35.5 | °C/W | Conforming to JESD51-7 (4 layers) |
| | Thermal Characterization Parameter | ψjb | 26.9 | | |
| 100 pin LQFP | Thermal Resistance | Θja | 38.3 | °C/W | Conforming to JESD51-7 (4 layers) |
| | Thermal Characterization Parameter | ψjb | 28.1 | | |

**Note:** The thermal resistance and characterization parameters depend on the usage environment.

## 40.28.2 Board

Conforming to JESD51-7 (4 layers)

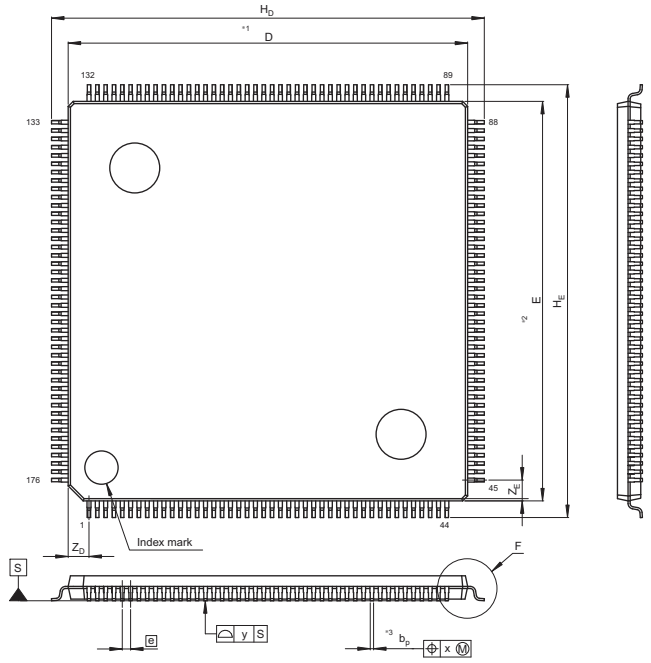
| | Board Size (mm) | | |
|---|---|---|---|
| | X | Y | Area ($mm^2$) |
| Board | 76.2 | 114.3 | 8709.66 |
| Remaining copper rates | Thickness of conductors | | |
| 50-95-95-50% | 70-35-35-70µm | | |

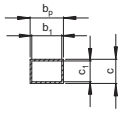
# Appendix A. Package

## A.1 Package Dimensions

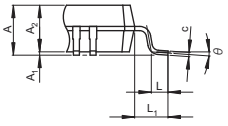
### A.1.1 176 Pin

| JEITA Package Code | RENESAS Code | Previous Code | MASS[Typ.] |
|---|---|---|---|
| P-LFQFP176-24x24-0.50 | PLQP0176KB-A | 176P6Q-A/FP-176E/FP-176EV | 1.8g |




Terminal cross section



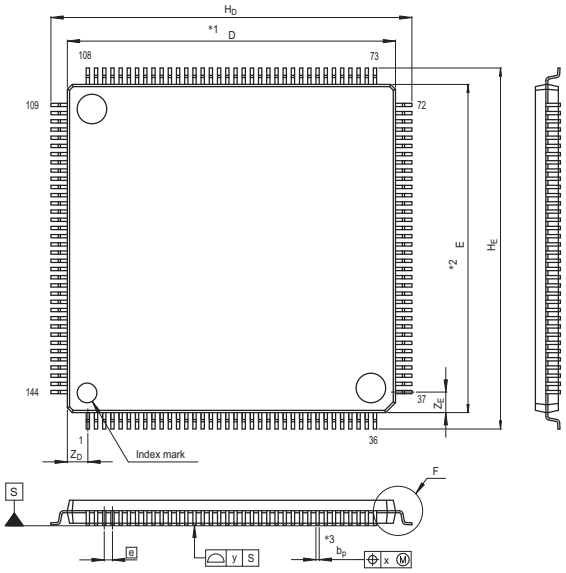

Detail F

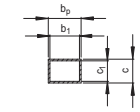
NOTE)
1. DIMENSIONS "*1" AND "*2" DO NOT INCLUDE MOLD FLASH.
2. DIMENSION "*3" DOES NOT INCLUDE TRIM OFFSET.

| Reference Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| D | 23.9 | 24.0 | 24.1 |
| E | 23.9 | 24.0 | 24.1 |
| $A_2$ | — | 1.4 | — |
| $H_D$ | 25.8 | 26.0 | 26.2 |
| $H_E$ | 25.8 | 26.0 | 26.2 |
| A | — | — | 1.7 |
| $A_1$ | 0.05 | 0.1 | 0.15 |
| $b_p$ | 0.15 | 0.20 | 0.25 |
| $b_1$ | — | 0.18 | — |
| c | 0.09 | 0.145 | 0.20 |
| $c_1$ | — | 0.125 | — |
| θ | 0° | — | 8° |
| e | — | 0.5 | — |
| x | — | — | 0.08 |
| y | — | — | 0.10 |
| $Z_D$ | — | 1.25 | — |
| $Z_E$ | — | 1.25 | — |
| L | 0.35 | 0.5 | 0.65 |
| $L_1$ | — | 1.0 | — |

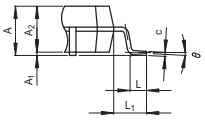
## A.1.2 144 Pin

| JEITA Package Code | RENESAS Code | Previous Code | MASS[Typ.] |
|---|---|---|---|
| P-LFQFP144-20x20-0.50 | PLQP0144KA-A | 144P6Q-A / FP-144L / FP-144LV | 1.2g |




Terminal cross section



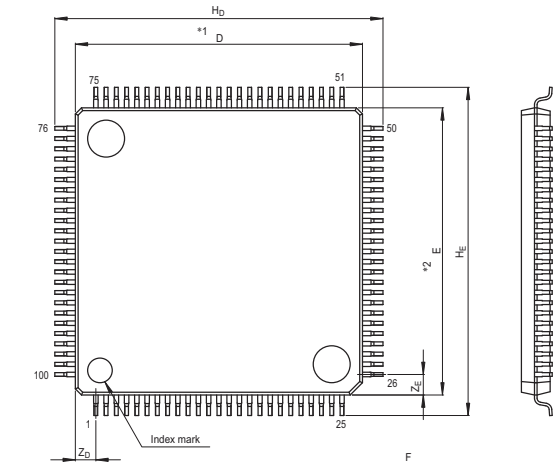

Detail F

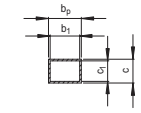
NOTE)
1. DIMENSIONS "*1" AND "*2" DO NOT INCLUDE MOLD FLASH.
2. DIMENSION "*3" DOES NOT INCLUDE TRIM OFFSET.

| Reference Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| D | 19.9 | 20.0 | 20.1 |
| E | 19.9 | 20.0 | 20.1 |
| $A_2$ | — | 1.4 | — |
| $H_D$ | 21.8 | 22.0 | 22.2 |
| $H_E$ | 21.8 | 22.0 | 22.2 |
| A | — | — | 1.7 |
| $A_1$ | 0.05 | 0.1 | 0.15 |
| $b_p$ | 0.17 | 0.22 | 0.27 |
| $b_1$ | — | 0.20 | — |
| c | 0.09 | 0.145 | 0.20 |
| $c_1$ | — | 0.125 | — |
| θ | 0° | — | 8° |
| e | — | 0.5 | — |
| x | — | — | 0.08 |
| y | — | — | 0.10 |
| $Z_D$ | — | 1.25 | — |
| $Z_E$ | — | 1.25 | — |
| L | 0.35 | 0.5 | 0.65 |
| $L_1$ | — | 1.0 | — |

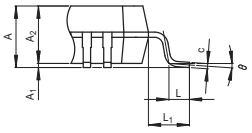
## A.1.3 100 Pin

| JEITA Package Code | RENESAS Code | Previous Code | MASS[Typ.] |
|---|---|---|---|
| P-LFQFP100-14x14-0.50 | PLQP0100KB-A | 100P6Q-A / FP-100U / FP-100UV | 0.6g |




Terminal cross section




Detail F

NOTE)
1. DIMENSIONS "*1" AND "*2" DO NOT INCLUDE MOLD FLASH.
2. DIMENSION "*3" DOES NOT INCLUDE TRIM OFFSET.

| Reference Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| D | 13.9 | 14.0 | 14.1 |
| E | 13.9 | 14.0 | 14.1 |
| $A_2$ | — | 1.4 | — |
| $H_D$ | 15.8 | 16.0 | 16.2 |
| $H_E$ | 15.8 | 16.0 | 16.2 |
| A | — | — | 1.7 |
| $A_1$ | 0.05 | 0.1 | 0.15 |
| $b_p$ | 0.15 | 0.20 | 0.25 |
| $b_1$ | — | 0.18 | — |
| c | 0.09 | 0.145 | 0.20 |
| $c_1$ | — | 0.125 | — |
| $\theta$ | 0° | — | 8° |
| e | — | 0.5 | — |
| x | — | — | 0.08 |
| y | — | — | 0.08 |
| $Z_D$ | — | 1.0 | — |
| $Z_E$ | — | 1.0 | — |
| L | 0.35 | 0.5 | 0.65 |
| $L_1$ | — | 1.0 | — |

RH850/F1K User's Manual: Hardware

Publication Date: Rev.0.50 Mar 31, 2015
Rev.1.10 Nov 30, 2016

Published by: Renesas Electronics Corporation

**SALES OFFICES** Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.


**Renesas Electronics America Inc.**
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

**Renesas Electronics Canada Limited**
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

**Renesas Electronics Europe Limited**
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

**Renesas Electronics Europe GmbH**
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

**Renesas Electronics (China) Co., Ltd.**
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

**Renesas Electronics (Shanghai) Co., Ltd.**
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

**Renesas Electronics Hong Kong Limited**
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

**Renesas Electronics Taiwan Co., Ltd.**
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

**Renesas Electronics Singapore Pte. Ltd.**
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

**Renesas Electronics Malaysia Sdn.Bhd.**
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

**Renesas Electronics India Pvt. Ltd.**
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

**Renesas Electronics Korea Co., Ltd.**
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141



© 2016 Renesas Electronics Corporation. All rights reserved.

Colophon 4.0

RH850/F1K

Renesas Electronics Corporation

R01UH0562EJ0110

# Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Renesas Electronics:

R7F7015873AFP-C#AA3 R7F7015814AFP-C#AA3 R7F7015834AFP-C#AA3 R7F7015874AFP-C#AA3 R7F7015834AFP-C#KA3 R7F7015834AFP-C#BA3